In [7]:

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

In [8]:

```
loans = pd.read_csv('loan_data.csv')
```

In [9]:

```
loans.head()
```

Out[9]:

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths
0	1	debt_consolidation	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	52.1	0
1	1	credit_card	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	76.7	0
2	1	debt_consolidation	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	25.6	1
3	1	debt_consolidation	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	73.2	1
4	1	credit_card	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	39.5	0

In [10]:

```
loans.shape
```

Out[10]:

```
(9578, 14)
```

In [11]:

```
loans['not.fully.paid'].value_counts()
```

Out[11]:

```
0    8045
1    1533
Name: not.fully.paid, dtype: int64
```

In [12]:

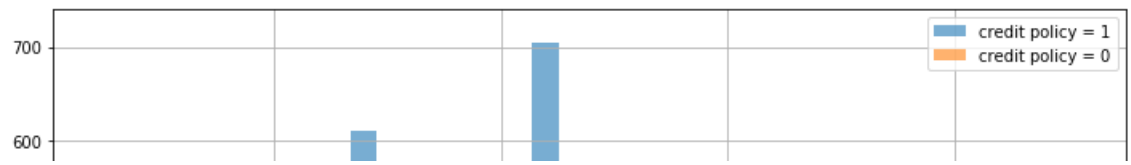
```
loans['purpose'].value_counts()
```

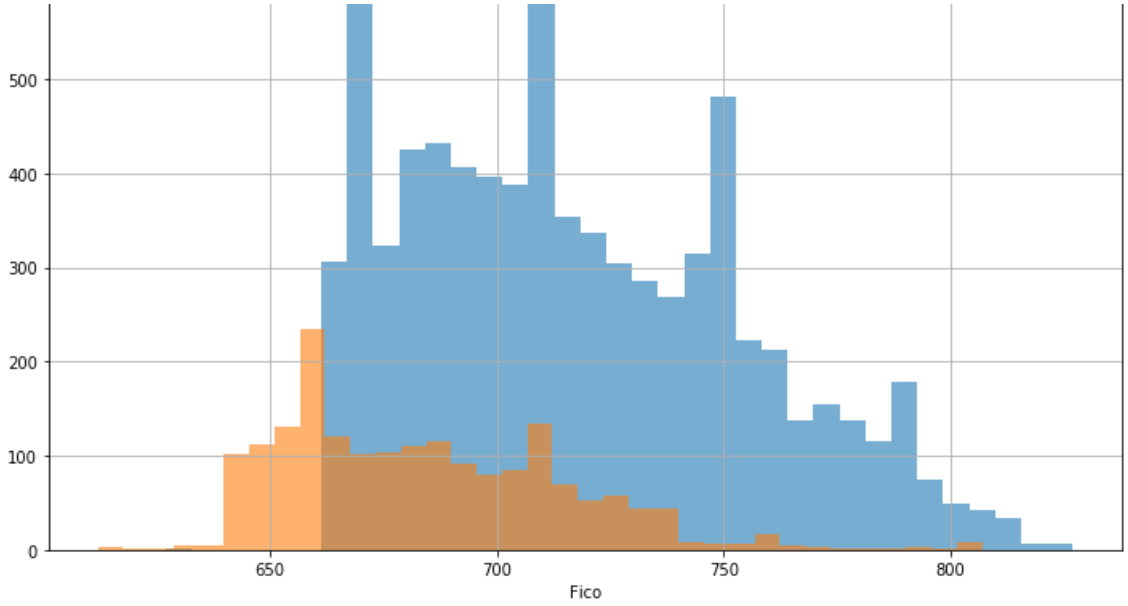
Out[12]:

```
debt_consolidation    3957
all_other             2331
credit_card           1262
home_improvement       629
small_business         619
major_purchase         437
educational            343
Name: purpose, dtype: int64
```

In [18]:

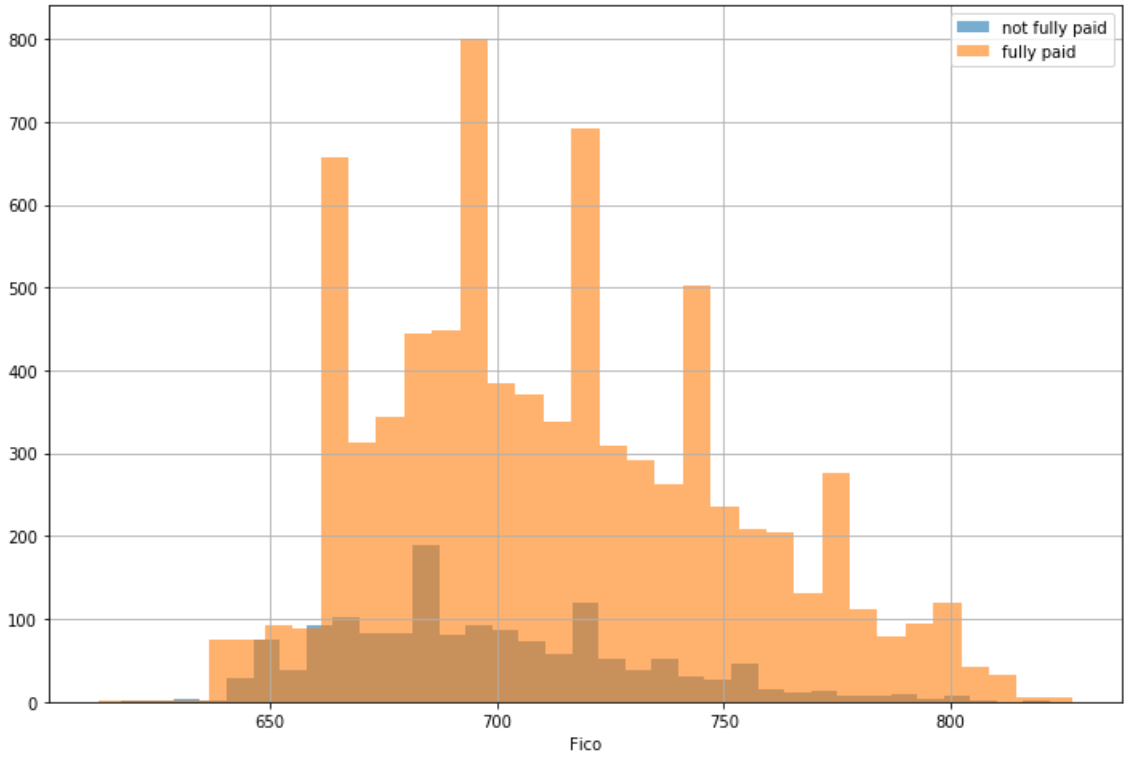
```
plt.figure(figsize=(12, 8))
loans[loans['credit.policy']==1]['fico'].hist(bins=35, label='credit policy = 1', alpha=0.6)
loans[loans['credit.policy']==0]['fico'].hist(bins=35, label='credit policy = 0', alpha=0.6)
plt.xlabel('Fico')
plt.legend();
```





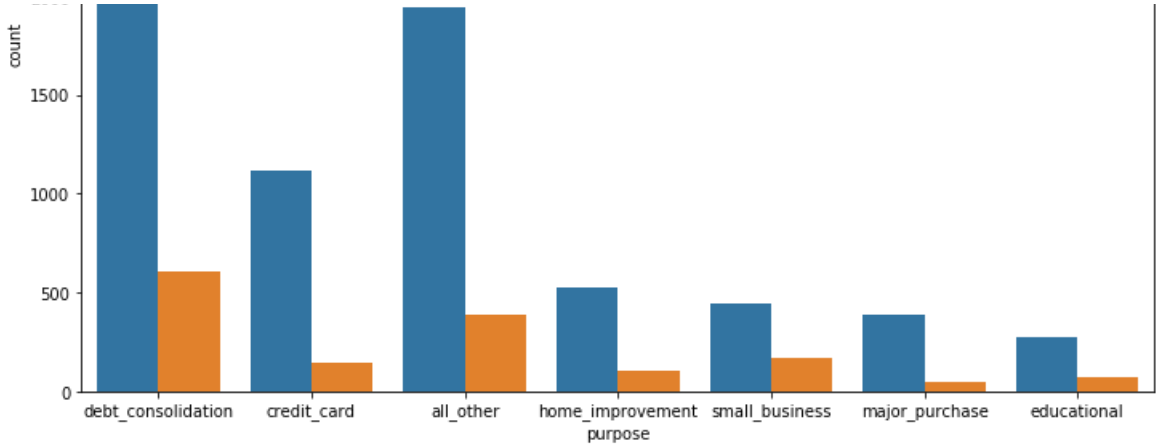
In [22]:

```
plt.figure(figsize=(12, 8))
loans[loans['not.fully.paid']==1]['fico'].hist(bins=35, label='not fully paid', alpha=0.6)
loans[loans['not.fully.paid']==0]['fico'].hist(bins=35, label='fully paid', alpha=0.6)
plt.xlabel('Fico')
plt.legend();
```



In [23]:

```
plt.figure(figsize=(12, 8))
sns.countplot(x='purpose', hue='not.fully.paid', data=loans);
```

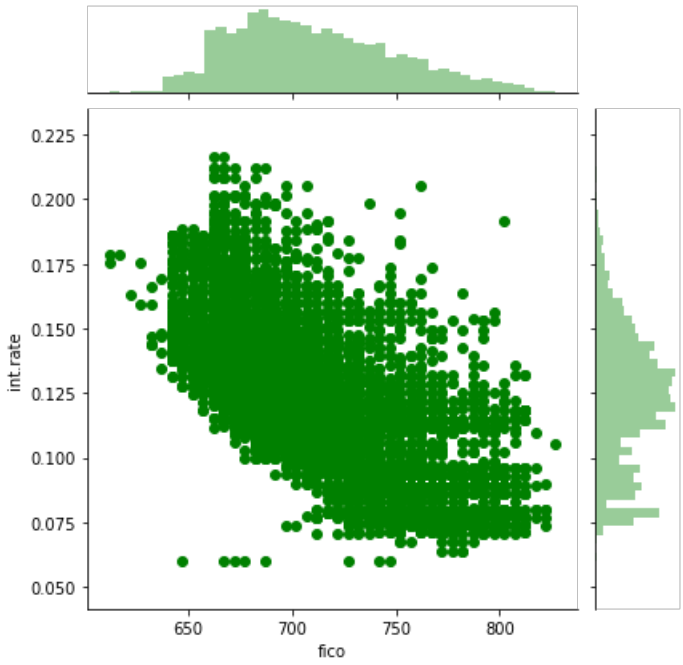


In [24]:

```
plt.figure(figsize=(12, 8))
sns.jointplot(x='fico', y='int.rate', data=loans, color='green');
```

```
/Users/sudeng/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

```
<Figure size 864x576 with 0 Axes>
```



In [25]:

```
cat_feats = ['purpose']
```

In [27]:

```
final_data = pd.get_dummies(loans, columns=cat_feats, drop_first=True)
```

In [28]:

```
final_data.head()
```

Out[28]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delinq.2yrs	pub.rec
0	1	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	52.1	0	0	0
1	1	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	76.7	0	0	0
2	1	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	25.6	1	0	0
3	1	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	73.2	1	0	0

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delinq.2yrs	pub.rec
4	1	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	39.5	0	1	0

In [29]:

```
from sklearn.cross_validation import train_test_split
```

In [36]:

```
x = final_data.drop('not.fully.paid', axis=1)
y = final_data['not.fully.paid']
```

In [37]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=101)
```

In [38]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [39]:

```
dtree = DecisionTreeClassifier()
```

In [40]:

```
dtree.fit(x_train, y_train)
```

Out[40]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [41]:

```
predictions = dtree.predict(x_test)
```

In [42]:

```
from sklearn.metrics import classification_report, confusion_matrix
```

In [43]:

```
print(classification_report(y_test, predictions))
```

```
             precision    recall  f1-score   support

          0       0.86      0.82      0.84      2431
          1       0.20      0.24      0.22       443

avg / total       0.75      0.73      0.74      2874
```

In [44]:

```
print(confusion_matrix(y_test, predictions))
```

```
[[2000  431]
 [ 336  107]]
```

In [45]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [47]:

```
rfc = RandomForestClassifier(n_estimators=300)
```

In [48]:

```
rfc.fit(x_train, y_train)
```

Out[48]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=300, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [49]:

```
rfc_pred = rfc.predict(x_test)
```

In [50]:

```
print(classification_report(y_test, rfc_pred))
```

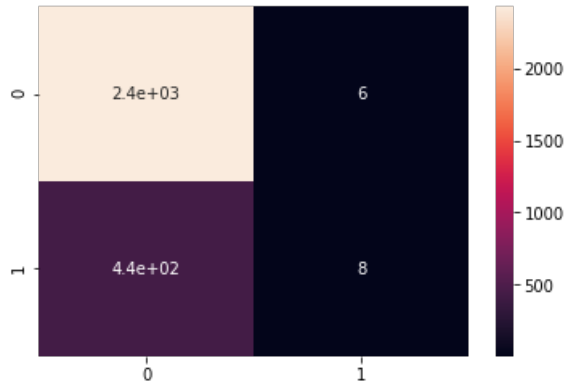
```
             precision    recall  f1-score   support

          0       0.85      1.00      0.92      2431
          1       0.57      0.02      0.04       443

avg / total       0.81      0.85      0.78      2874
```

In [51]:

```
cm = confusion_matrix(y_test, rfc_pred)
```

In [52]:

```
sns.heatmap(cm, annot=True);
```



In [ ]:

```

```