In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import patsy
import numpy as np
import seaborn as sns

import patsy                            # provides a syntax for specifyin
g models
import statsmodels.api as sm            # provides statistical models lik
e ols, gmm, anova, etc...
import statsmodels.formula.api as smf   # p

%matplotlib inline
```

load dataset and clean

In [2]:
```
Q108 = pd.read_csv("/Users/scottlai/Desktop/work/FreddieMacProjects/cod
e/historical_data1_Q12008/Q12008.csv",
                     thousands=',')
#Q108.head()

x18 = Q108.loc[(Q108['PROPERTY TYPE '] == 'SF')& (Q108['Credit Score'] >
=550) & (Q108['Credit Score'] <=850)]
#x18.head()

Q109 = pd.read_csv("/Users/scottlai/Desktop/work/FreddieMacProjects/cod
e/historical_data1_Q12009/Q109.csv",
                  thousands = ',')
x19 = Q109.loc[(Q109['PROPERTY TYPE '] == 'SF')& (Q109['Credit Score'] >
=550) & (Q109['Credit Score'] <=850)]

Q208 = pd.read_csv('/Users/scottlai/Desktop/work/FreddieMacProjects/cod
e/historical_data1_Q22008/Q208.csv',
                  thousands= ',')
x28 = Q208.loc[(Q208['PROPERTY TYPE '] == 'SF')& (Q208['Credit Score'] >
=550) & (Q208['Credit Score'] <=850)]

Q209 = pd.read_csv('/Users/scottlai/Desktop/work/FreddieMacProjects/cod
e/historical_data1_Q22009/Q209.csv',
                  thousands= ',')
x29 = Q209.loc[(Q209['PROPERTY TYPE '] == 'SF')& (Q209['Credit Score'] >
=550) & (Q209['Credit Score'] <=850)]

Q308 = pd.read_csv('/Users/scottlai/Desktop/work/FreddieMacProjects/cod
e/historical_data1_Q32008/Q308.csv',
                  thousands= ',')
x38 = Q308.loc[(Q308['PROPERTY TYPE '] == 'SF')& (Q308['Credit Score'] >
=550) & (Q308['Credit Score'] <=850)]

Q309 = pd.read_csv('/Users/scottlai/Desktop/work/FreddieMacProjects/cod
e/historical_data1_Q32009/Q309.csv',
                  thousands= ',')
x39 = Q309.loc[(Q309['PROPERTY TYPE '] == 'SF')& (Q309['Credit Score'] >
=550) & (Q309['Credit Score'] <=850)]

Q408 = pd.read_csv('/Users/scottlai/Desktop/work/FreddieMacProjects/cod
e/historical_data1_Q42008/Q408.csv',
                  thousands= ',')
x48 = Q408.loc[(Q408['PROPERTY TYPE '] == 'SF')& (Q408['Credit Score'] >
=550) & (Q408['Credit Score'] <=850)]

Q409 = pd.read_csv('/Users/scottlai/Desktop/work/FreddieMacProjects/cod
e/historical_data1_Q42009/Q409.csv',
                  thousands= ',')
x49 = Q409.loc[(Q409['PROPERTY TYPE '] == 'SF')& (Q409['Credit Score'] >
=550) & (Q409['Credit Score'] <=850)]
```

```
/Users/scottlai/anaconda3/lib/python3.6/site-packages/IPython/core/inte
ractiveshell.py:3058: DtypeWarning: Columns (25) have mixed types. Spec
ify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

merge the datasets

In [3]:
```
frame = [x18,x19,x28,x29,x38,x39,x48,x49]
data = pd.concat(frame)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2345307 entries, 0 to 350771
Data columns (total 27 columns):
Credit Score                                int64
Monthly report period                       int64
FIRST TIME HOMEBUYER FLAG                   object
CURRENT LOAN DELINQUENCY STATUS             int64
X5                                          float64
MORTGAGE INSURANCE PERCENTAGE (MI %)        int64
NUMBER OF UNITS                             int64
OCCUPANCY STATUS                            object
ORIGINAL COMBINED LOAN-TO-VALUE (CLTV)      int64
ORIGINAL DEBT-TO-INCOME (DTI) RATIO         int64
X11                                         int64
ORIGINAL LOAN-TO-VALUE (LTV)                int64
X13                                         float64
CHANNEL                                     object
PREPAYMENT PENALTY MORTGAGE (PPM) FLAG      object
PRODUCT TYPE                                object
X17                                         object
PROPERTY TYPE                               object
X19                                         float64
 Loan Sequence Number                       object
LOAN PURPOSE                                object
X22                                         int64
NUMBER OF BORROWERS                         int64
STEP MODIFICATION FLAG                      object
X25                                         object
X26                                         object
X27                                         object
dtypes: float64(3), int64(11), object(13)
memory usage: 501.0+ MB
```

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

Out[4]:

	Credit Score	Monthly report period	FIRST TIME HOMEBUYER FLAG	CURRENT LOAN DELINQUENCY STATUS	X5	MORTGAGE INSURANCE PERCENTAGE (MI %)	NUMBER OF UNITS	OCCUPANC STATU
0	771	200803	N	203802	NaN	0	1	
1	729	200805	N	203804	17140.0	0	1	
2	769	200803	N	203802	NaN	0	1	
3	755	200803	Y	203802	NaN	35	1	
4	760	200804	N	203803	48300.0	0	1	

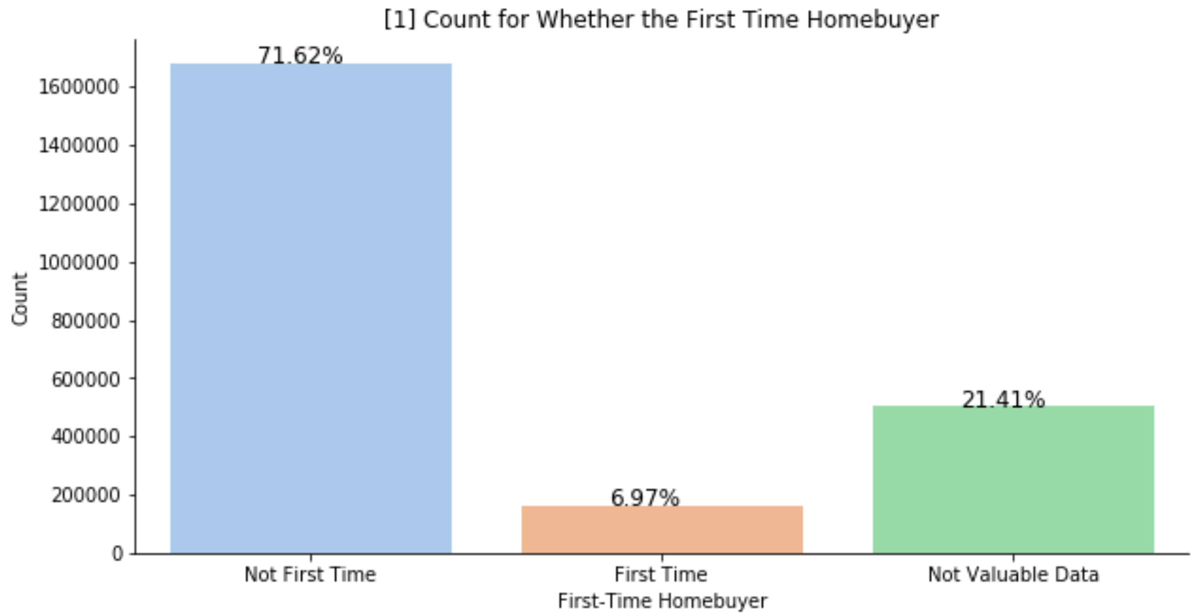
5 rows × 27 columns

Part 1: Exploratory Data Analysis

（1）单个基本数据分析

1. 是否为第一次购房

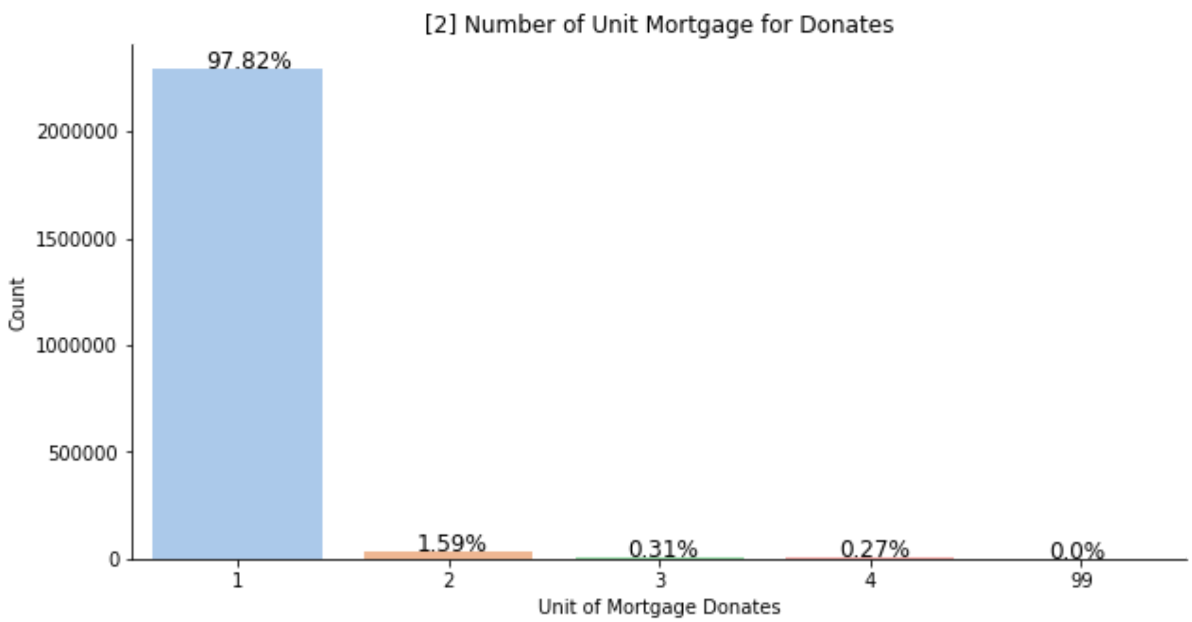
In [5]:
```
# 1）是否为第一次购房

data['FIRST TIME HOMEBUYER FLAG '].replace('N', 'Not First Time',inplace
= True)
data['FIRST TIME HOMEBUYER FLAG '].replace('Y', 'First Time',inplace = T
rue)
data['FIRST TIME HOMEBUYER FLAG '].replace('9', 'Not Valuable Data',inpl
ace = True)

fig, firsthouse = plt.subplots(figsize=(10,5))


sns.set_palette('pastel')
sns.countplot(x=data['FIRST TIME HOMEBUYER FLAG '], data=data)
data

firsthouse.set_title('[1] Count for Whether the First Time Homebuyer')
firsthouse.set_ylabel('Count')
firsthouse.set_xlabel('First-Time Homebuyer')

firsthouse.spines['right'].set_visible(False) # get ride of the line on
 the right
firsthouse.spines['top'].set_visible(False)

totals = []

for i in firsthouse.patches:
    totals.append(i.get_height())


total = sum(totals)

for i in firsthouse.patches:
    firsthouse.text(i.get_x() +.25,i.get_height()+20,\
            str(round((i.get_height()/total)*100,2))+"%",fontsize = 12,
            color = "black")
plt.show()
```



上图 【1】显示借贷者个人或群体是否为第一次购买房屋的分布数据，根据图像显示，有近71.62%的借贷者不是第一次购买房屋，而只有6.97% 的借贷者为第一次购买房屋。而数据有接近21.41%的无效数据。

2. 用多少房子做抵押

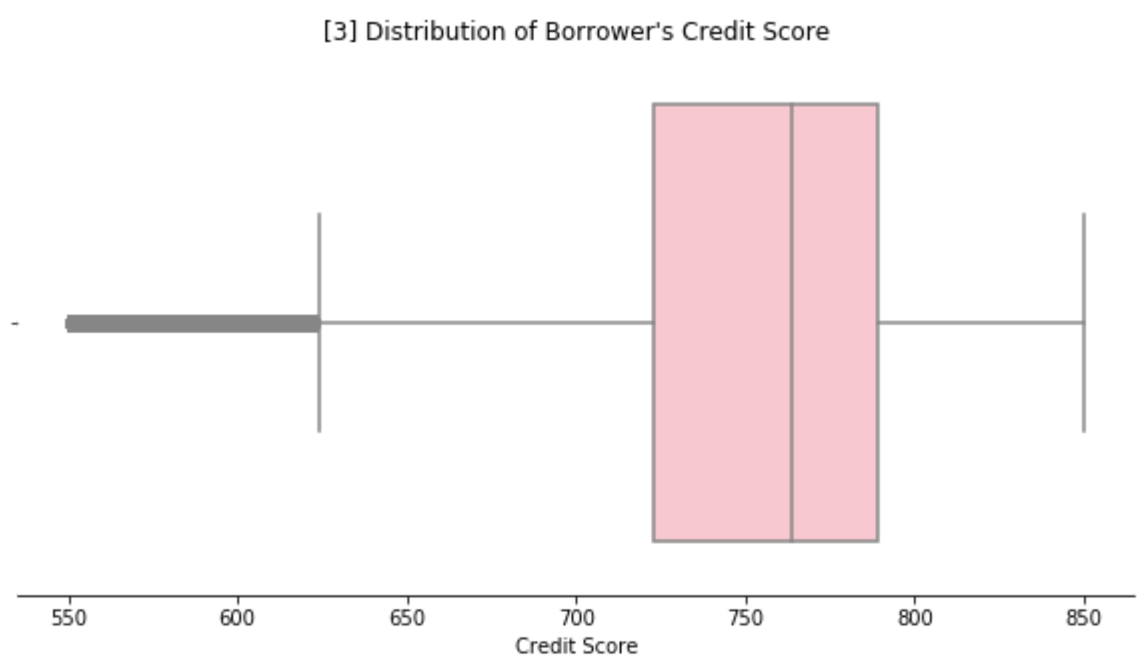
In [6]:
```
fig, firsthouse = plt.subplots(figsize=(10,5))


sns.set_palette('pastel')
sns.countplot(x=data['NUMBER OF UNITS '], data=data)
data

firsthouse.set_title('[2] Number of Unit Mortgage for Donates')
firsthouse.set_ylabel('Count')
firsthouse.set_xlabel('Unit of Mortgage Donates')


firsthouse.spines['right'].set_visible(False) # get ride of the line on
 the right
firsthouse.spines['top'].set_visible(False)

totals = []

for i in firsthouse.patches:
    totals.append(i.get_height())


total = sum(totals)

for i in firsthouse.patches:
    firsthouse.text(i.get_x() +.25,i.get_height()+20,\
            str(round((i.get_height()/total)*100,2))+"%",fontsize = 12,
            color = "black")
plt.show()
```



从图【2】中可看出97.82%的借贷着只有一座房子作为贷款抵押。

3. 信用分数基本分布

In [7]:
```
fig, FICO = plt.subplots(figsize=(10,5))
sns.set_palette('dark')
sns.boxplot(x = data['Credit Score'],color = 'pink')


FICO.set_title("[3] Distribution of Borrower's Credit Score")

FICO.set_xlabel('Credit Score')


FICO.spines['right'].set_visible(False) # get ride of the line on the ri
ght
FICO.spines['top'].set_visible(False)
FICO.spines['left'].set_visible(False)
totals = []

for i in firsthouse.patches:
    totals.append(i.get_height())


total = sum(totals)

for i in firsthouse.patches:
    firsthouse.text(i.get_x() +.25,i.get_height()+20,\
            str(round((i.get_height()/total)*100,2))+"%",fontsize = 12,
            color = "black")
plt.show()
```



In [8]:
```
data['Credit Score'].describe()
```

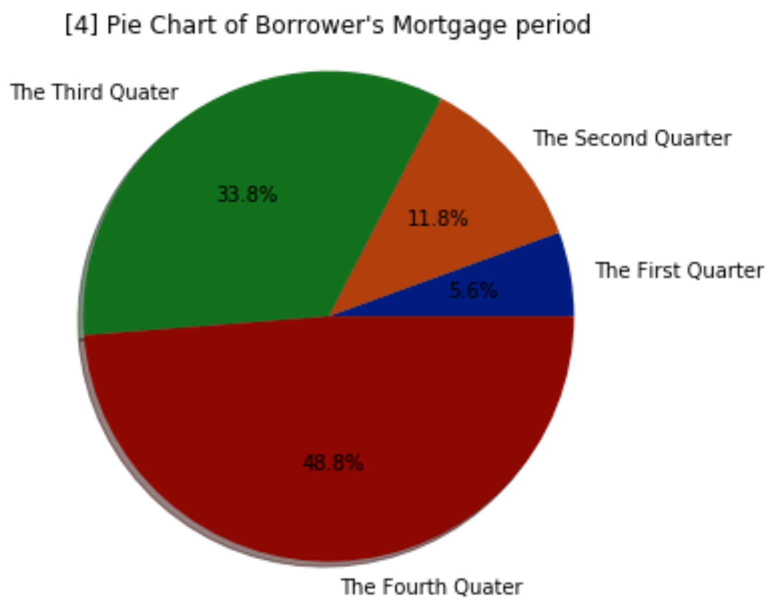
Out[8]:
```
count    2.345307e+06
mean     7.523151e+02
std      4.651327e+01
min      5.500000e+02
25%      7.230000e+02
50%      7.640000e+02
75%      7.890000e+02
max      8.500000e+02
Name: Credit Score, dtype: float64
```

图【3】是关于借贷人信用分数的分布数据图，结合箱形图及数据概括，我们可以看出，借贷人的平均信用分数为752分左右，大部分人（25%-75%）的信用分数集中在723到789分之间，贷款人信用分最低为550分，最高接近850.

4. 好奇哪个季度来借贷的人最多

In [9]:
```
#data['Monthly report period'].replace({200803:3,200903:3,200804:4,200904:4,
#                                       200801:1,200901:1,200802:2,200902:2},inplace = True)
```

In [10]:
```
#(data['Monthly report period']== 1).sum() #31490
#(data['Monthly report period']== 2).sum() #67074
#(data['Monthly report period']== 3).sum() #191467
#(data['Monthly report period']== 4).sum() #276164
sizes = [31490,67074,191467,276164]

labels = ['The First Quarter', 'The Second Quarter','The Third Quater',
'The Fourth Quater']

fig, ax = plt.subplots(figsize=(10,5))

sns.set_palette('pastel')
ax.pie(sizes,labels = labels,autopct = '%1.1f%%',shadow = True)
ax.axis('equal')


ax.set_title("[4] Pie Chart of Borrower's Mortgage period")

plt.show()
```



从图【4】中可以看出有48.8%的借贷者都在第四季度（9-12月）间进行借贷。第一季度借贷人数最少，只有5.6%

(2) 数据变量间关系分析

In [11]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2345307 entries, 0 to 350771
Data columns (total 27 columns):
Credit Score                                 int64
Monthly report period                        int64
FIRST TIME HOMEBUYER FLAG                    object
CURRENT LOAN DELINQUENCY STATUS              int64
X5                                           float64
MORTGAGE INSURANCE PERCENTAGE (MI %)         int64
NUMBER OF UNITS                              int64
OCCUPANCY STATUS                             object
ORIGINAL COMBINED LOAN-TO-VALUE (CLTV)       int64
ORIGINAL DEBT-TO-INCOME (DTI) RATIO          int64
X11                                          int64
ORIGINAL LOAN-TO-VALUE (LTV)                 int64
X13                                          float64
CHANNEL                                      object
PREPAYMENT PENALTY MORTGAGE (PPM) FLAG       object
PRODUCT TYPE                                 object
X17                                          object
PROPERTY TYPE                                object
X19                                          float64
 Loan Sequence Number                        object
LOAN PURPOSE                                 object
X22                                          int64
NUMBER OF BORROWERS                          int64
STEP MODIFICATION FLAG                       object
X25                                          object
X26                                          object
X27                                          object
dtypes: float64(3), int64(11), object(13)
memory usage: 541.0+ MB
```

根据数据具体分析，我们可以发现自变量分为数值变量及单位变量，可用做观测值的数值变量为$CLTV$[1]，DTI[2]，及LTV[3]， 而可用做单位变量观测值的则有信用分数（Credit Score），时间（Monthly Report Period），渠道（Channel）。

首先对变量进行线性回归测试

我们首先考虑基本线性模型

```
y ~ x1 + x2
```

概括方程式：

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon.$$

（1）如果假设应变量(y) 为CLTV，自变量则为时间(x_1)，信用分数(x_2)及渠道(x_3)。

则有：

$$CLTV = \beta_0 + \beta_1 Time + \beta_2 CreditScore + \beta_3 Channel + \epsilon.$$

In [12]:
```
data.rename(columns={'ORIGINAL DEBT-TO-INCOME (DTI) RATIO ':'DTI','ORIGINAL LOAN-TO-VALUE (LTV) ':'LTV'}, inplace=True)
data.rename(columns={'ORIGINAL COMBINED LOAN-TO-VALUE (CLTV) ':'CLTV','Monthly report period':'Time','Credit Score':'CS','CHANNEL ':'Channel'}, inplace=True)
```

In [13]:
```
data = data[data['CLTV'] != 999]
data = data[data['DTI'] != 999]
data = data[data['LTV'] != 999]
```

In [14]:
```
y, X = patsy.dmatrices('CLTV~ Time + CS + Channel', data)
cltv = sm.OLS(y, X)
res = cltv.fit()
print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:                   CLTV   R-squared:
0.033
Model:                            OLS   Adj. R-squared:
0.033
Method:                 Least Squares   F-statistic:                     1.
580e+04
Date:                Thu, 22 Aug 2019   Prob (F-statistic):
0.00
Time:                        00:13:53   Log-Likelihood:                -9.8
906e+06
No. Observations:             2321634   AIC:                             1.
978e+07
Df Residuals:                 2321628   BIC:                             1.
978e+07
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
=========
                   coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
---------
Intercept     4268.5975     44.610     95.686      0.000    4181.163
4356.032
Channel[T.C]     0.7845      0.038     20.578      0.000       0.710
0.859
Channel[T.R]    -1.6742      0.034    -49.717      0.000      -1.740
-1.608
Channel[T.T]     0.9911      0.051     19.573      0.000       0.892
1.090
Time            -0.0207      0.000    -93.249      0.000      -0.021
-0.020
CS              -0.0500      0.000   -201.056      0.000      -0.050
-0.049
==============================================================================
=======
Omnibus:                   182661.288   Durbin-Watson:
1.755
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              229
766.234
Skew:                          -0.769   Prob(JB):
0.00
Kurtosis:                       3.084   Cond. No.
7.97e+08
==============================================================================
=======

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 7.97e+08. This might indicate that t
here are
strong multicollinearity or other numerical problems.
```

In [15]:
```
print('The parameters are:', res.params, '\n')
print('The confidence intervals are:', res.conf_int(), '\n')
print('The r-sqared is:', res.rsquared)
```

```
The parameters are: [ 4.26859750e+03  7.84504077e-01 -1.67417997e+00
9.91054955e-01
 -2.07209804e-02 -4.99568964e-02]

The confidence intervals are: [[ 4.18116256e+03  4.35603244e+03]
 [ 7.09783907e-01  8.59224246e-01]
 [-1.74017977e+00 -1.60818018e+00]
 [ 8.91814557e-01  1.09029535e+00]
 [-2.11565084e-02 -2.02854524e-02]
 [-5.04438946e-02 -4.94698982e-02]]

The r-sqared is: 0.03291321655997914
```

从上表线性回归测试我们可以看出原始组合贷款价值（CLTV）与渠道的变化关系较为明显，其中通过零售商(T.R)渠道进行借贷了解与CLTV数值成反比，及通过零售进行借贷了解会造成CLTV数值降低（-1.6742）。反之，通过通信人（T.C)或未表明具体渠道了解借贷渠道则会使CLTV数值升高。同理，CLTV与时间及信用值成反比。

（2）如果假设应变量(y) 为DTI，自变量则为时间(x_1)**，信用分数**(x_2)**及渠道**(x_3)**。**

则有：

$$DTI = \beta_0 + \beta_1 Time + \beta_2 CreditScore + \beta_3 Channel + \epsilon.$$

In [16]:
```
y, X = patsy.dmatrices('DTI~ Time + CS + Channel', data)
DTI = sm.OLS(y, X)
res1 = DTI.fit()
print(res1.summary())
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:                    DTI   R-squared:
0.082
Model:                            OLS   Adj. R-squared:
0.082
Method:                 Least Squares   F-statistic:                      4.
121e+04
Date:                Thu, 22 Aug 2019   Prob (F-statistic):
0.00
Time:                        00:14:00   Log-Likelihood:                -9.0
177e+06
No. Observations:             2321634   AIC:                              1.
804e+07
Df Residuals:                 2321628   BIC:                              1.
804e+07
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
=========
                   coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
---------
Intercept     4114.1671     30.630    134.318      0.000    4054.133
4174.201
Channel[T.C]    -1.0113      0.026    -38.635      0.000      -1.063
-0.960
Channel[T.R]    -3.0740      0.023   -132.955      0.000      -3.119
-3.029
Channel[T.T]     0.0819      0.035      2.355      0.019       0.014
0.150
Time            -0.0201      0.000   -131.700      0.000      -0.020
-0.020
CS              -0.0566      0.000   -331.623      0.000      -0.057
-0.056
==============================================================================
=======
Omnibus:                    48653.482   Durbin-Watson:
1.898
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                29
975.064
Skew:                           0.133   Prob(JB):
0.00
Kurtosis:                       2.511   Cond. No.
7.97e+08
==============================================================================
=======

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 7.97e+08. This might indicate that t
here are
strong multicollinearity or other numerical problems.
```

In [17]:
```
print('The parameters are:', res1.params, '\n')
print('The confidence intervals are:', res1.conf_int(), '\n')
print('The r-sqared is:', res1.rsquared)
```

```
The parameters are: [ 4.11416706e+03 -1.01130710e+00 -3.07404107e+00
8.18597165e-02
 -2.00938595e-02 -5.65762535e-02]

The confidence intervals are: [[ 4.05413330e+03  4.17420082e+03]
 [-1.06261076e+00 -9.60003428e-01]
 [-3.11935724e+00 -3.02872490e+00]
 [ 1.37201985e-02  1.49999234e-01]
 [-2.03928977e-02 -1.97948213e-02]
 [-5.69106317e-02 -5.62418753e-02]]

The r-sqared is: 0.08151451332404758
```

从上表线性回归测试我们可以看出原始债务收入（DTI）比率与渠道的变化关系较为明显，其中通过零售商(T.R)渠道及通过通信人（T.C)渠道进行借贷了解与CLTV数值成反比，及通过零售进行借贷了解会造成CLTV数值降低（-3.0740 & -1.0113）。反之，未表明具体渠道了解借贷渠道则会使CLTV数值升高。同理，CLTV与时间及信用值成反比。

（3）如果假设应变量(y) 为LTV，自变量则为时间(x_1)**，信用分数**(x_2)**及渠道**(x_3)**。**

则有：

$$LTV = \beta_0 + \beta_1 Time + \beta_2 CreditScore + \beta_3 Channel + \epsilon.$$

In [18]:
```
y, X = patsy.dmatrices('LTV~ Time + CS + Channel', data)
ltv = sm.OLS(y, X)
res2 = ltv.fit()
print(res2.summary())
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:                    LTV   R-squared:
0.036
Model:                            OLS   Adj. R-squared:
0.036
Method:                 Least Squares   F-statistic:                     1.
729e+04
Date:                Thu, 22 Aug 2019   Prob (F-statistic):
0.00
Time:                        00:14:08   Log-Likelihood:                -9.8
922e+06
No. Observations:             2321634   AIC:                             1.
978e+07
Df Residuals:                 2321628   BIC:                             1.
978e+07
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
=========
                   coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
---------
Intercept     4325.2867     44.641     96.891      0.000    4237.793
4412.781
Channel[T.C]     0.7098      0.038     18.605      0.000       0.635
0.785
Channel[T.R]    -2.0986      0.034    -62.278      0.000      -2.165
-2.033
Channel[T.T]     0.8186      0.051     16.156      0.000       0.719
0.918
Time            -0.0210      0.000    -94.447      0.000      -0.021
-0.021
CS              -0.0518      0.000   -208.533      0.000      -0.052
-0.051
==============================================================================
=======
Omnibus:                   168682.301   Durbin-Watson:
1.749
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              208
805.110
Skew:                          -0.735   Prob(JB):
0.00
Kurtosis:                       2.993   Cond. No.
7.97e+08
==============================================================================
=======

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 7.97e+08. This might indicate that t
here are
strong multicollinearity or other numerical problems.
```

In [19]:
```
print('The parameters are:', res2.params, '\n')
print('The confidence intervals are:', res2.conf_int(), '\n')
print('The r-sqared is:', res2.rsquared)
```

```
The parameters are: [ 4.32528666e+03  7.09766242e-01 -2.09855253e+00
8.18573303e-01
 -2.10014238e-02 -5.18497814e-02]

The confidence intervals are: [[ 4.23779260e+03  4.41278072e+03]
 [ 6.34995547e-01  7.84536937e-01]
 [-2.16459696e+00 -2.03250810e+00]
 [ 7.19265800e-01  9.17880806e-01]
 [-2.14372463e-02 -2.05656013e-02]
 [-5.23371089e-02 -5.13624539e-02]]

The r-sqared is: 0.035895959086448936
```

从上表线性回归测试我们可以看出原始贷款价值（LTV）与渠道的变化关系较为明显，其中通过零售商(T.R)渠道渠道进行借贷了解与CLTV数值成反比，及通过零售进行借贷了解会造成CLTV数值降低（-2.0986）。反之，及通过通信人（T.C)及未表明具体渠道了解借贷渠道则会使CLTV数值升高。同理，CLTV与时间及信用值成反比。

（4）如果假设应变量(y) 为LTV，自变量则为CLTV(x_1)**，DTI**(x_2)**。**

则有：

$$LTV = \beta_0 + \beta_1 CLTV + \beta_2 DTI + \epsilon.$$

In [20]:
```
y, X = patsy.dmatrices('LTV~ CLTV + DTI', data)
ae = sm.OLS(y, X)
res3 = ae.fit()
print(res3.summary())
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:                    LTV   R-squared:
0.922
Model:                            OLS   Adj. R-squared:
0.922
Method:                 Least Squares   F-statistic:                       1.
372e+07
Date:                Thu, 22 Aug 2019   Prob (F-statistic):
0.00
Time:                        00:14:09   Log-Likelihood:                  -6.9
734e+06
No. Observations:             2321634   AIC:                               1.
395e+07
Df Residuals:                 2321631   BIC:                               1.
395e+07
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
=======
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-------
Intercept      1.0236      0.015     70.457      0.000       0.995
1.052
CLTV           0.9614      0.000   5162.058      0.000       0.961
0.962
DTI            0.0081      0.000     30.485      0.000       0.008
0.009
==============================================================================
=======
Omnibus:                  2202937.612   Durbin-Watson:
1.908
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               84572
632.133
Skew:                          -4.750   Prob(JB):
0.00
Kurtosis:                      31.000   Cond. No.
352.
==============================================================================
=======

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

In [21]:
```
print('The parameters are:', res3.params, '\n')
print('The confidence intervals are:', res3.conf_int(), '\n')
print('The r-sqared is:', res3.rsquared)
```

```
The parameters are: [1.02361275 0.96139035 0.00805852]

The confidence intervals are: [[0.99513819 1.05208731]
 [0.96102532 0.96175537]
 [0.00754041 0.00857663]]

The r-sqared is: 0.9219940608311492
```

从上表线性回归测试我们可以看出原始贷款价值（LTV）与原始债务收入（DTI）比率及原始组合贷款价值（CLTV）的关系都成正比。

相关关系作图

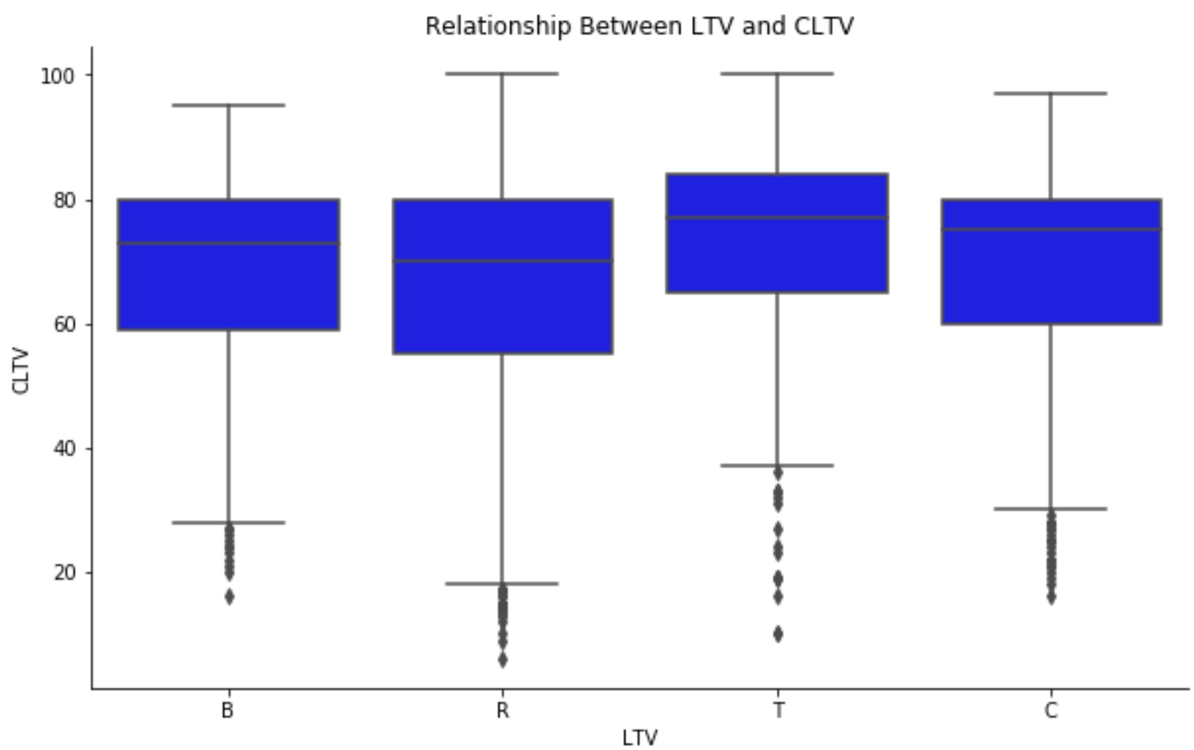
In [22]:
```
dataq = data.sample(n = 1000)
dataq.head()
```

Out[22]:

	CS	Time	FIRST TIME HOMEBUYER FLAG	CURRENT LOAN DELINQUENCY STATUS	X5	MORTGAGE INSURANCE PERCENTAGE (MI %)	NUMBER OF UNITS	OCCUPAI STA
130516	781	200902	First Time	203901	39100.0	0	1	
11372	729	200803	Not First Time	203802	26900.0	25	1	
656	773	200903	Not First Time	203902	NaN	0	1	
140984	757	201002	Not First Time	204001	25420.0	0	1	
84942	777	200804	Not First Time	203803	39580.0	0	1	

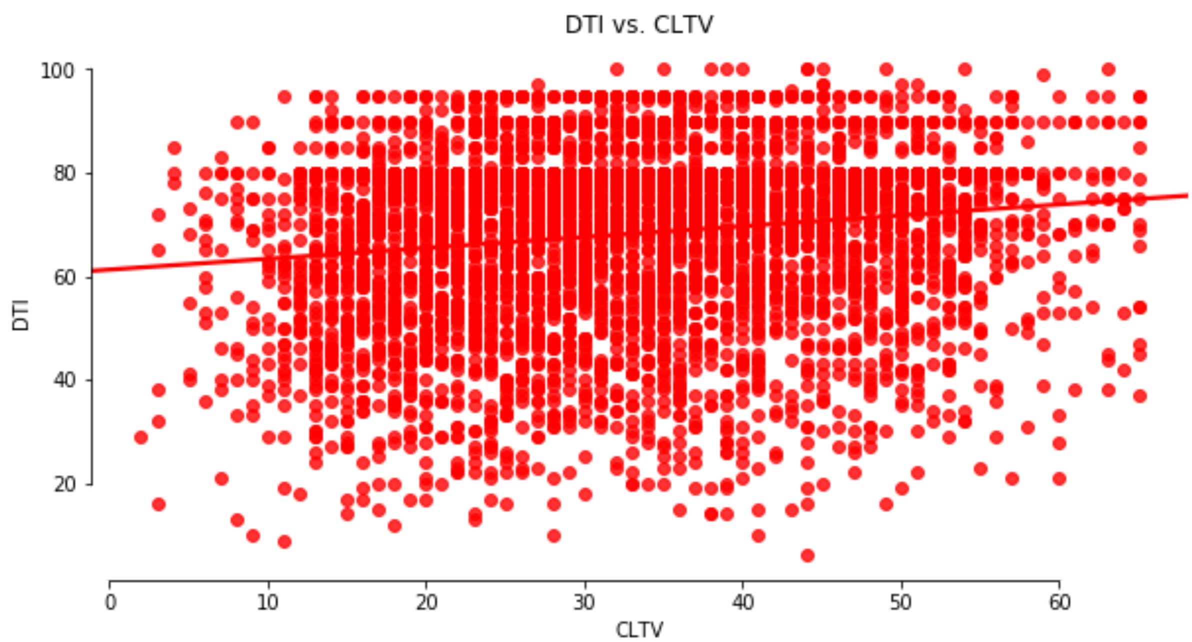
5 rows × 27 columns

In [23]:
```
dataq = data.sample(n = 5000)

fig, ax = plt.subplots(figsize=(10,6))

#sns.despine(fig,left = True, bottom = True)
clarity_ranking = ['R','B','C','T',9]
sns.boxplot(dataq['Channel'], dataq['CLTV'], color='blue', hue_order = c
larity_ranking )  # s is marker size

ax.set_title('Relationship Between LTV and CLTV')
ax.set_ylabel('CLTV')
ax.set_xlabel('LTV')

# Add some text to the figure
#ax.text(10, 2, 'A positive relationship!')

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

plt.show()
```



从上图可以看出原始贷款价值（LTV）每个分类与原始组合贷款价值（CLTV）的关系值。从中可以看出LTV（T）的CLTV平均值最高。

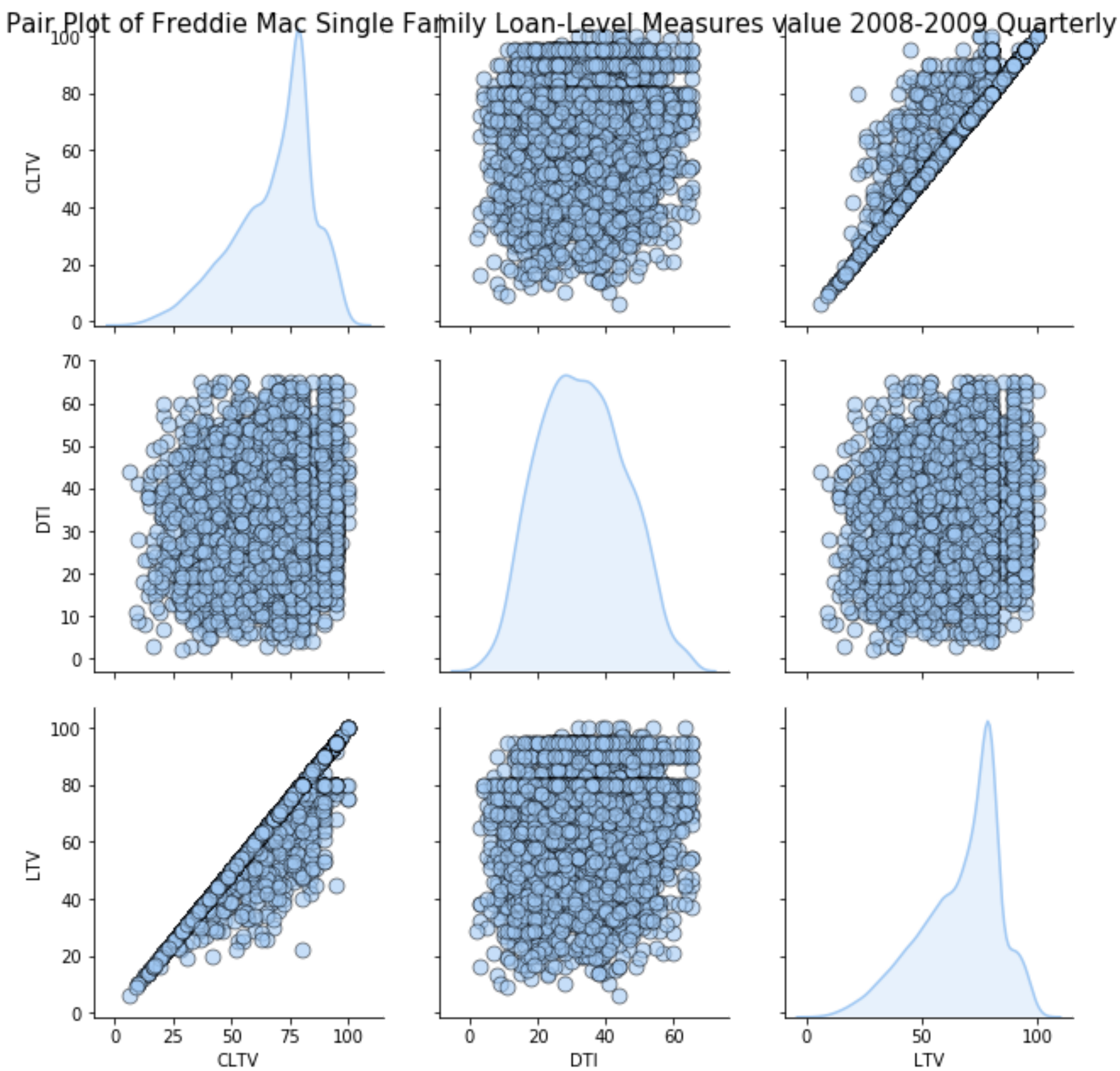
In [24]:
```
fig, ax = plt.subplots(figsize=(10,5))

# Theh seaborn regplot command. This is not called on an axis object (li
ke in matplotlib)
# but we can pass it an axis so that we can do matplotlib-like tweaking.
sns.regplot(x='DTI', y='CLTV', data=dataq,     # the data
            ax = ax,                                              # an a
xis object
            color = 'red',                                       # make
 it blue
            ci = 15)                                              # conf
idence interval: pass it the percent

sns.despine(ax = ax, trim=True)     # a bit easier than matplotlib
                                    # trim limits the axis to the data (v
ery Tufte-esque)

# Our usual labeling
ax.set_title('DTI vs. CLTV')
ax.set_ylabel('DTI')
ax.set_xlabel('CLTV')

plt.show()
```



从上图可以看出DTI和CLTV成正相关。

In [48]:
```
df = dataq.loc[:,['CLTV','DTI',"LTV"]]
sns.pairplot(df,diag_kind= 'kde', plot_kws = {'alpha':0.6,'s':80,'edgeco
lor':'k'},size = 3)
plt.suptitle('Pair Plot of Freddie Mac Single Family Loan-Level Measures
value 2008-2009 Quarterly',size = 15)
plt.show()
```

```
/Users/scottlai/anaconda3/lib/python3.6/site-packages/seaborn/axisgrid.
py:2065: UserWarning: The `size` parameter has been renamed to `height
`; pleaes update your code.
  warnings.warn(msg, UserWarning)
```



上图所展示了CLTV，DTI，及LTV三者的关系图，可见CLTV & DTI 和DTI & LTV的相关关系不明显，而CLTV & LTV关系明显为正向相关。同时可以看出三个数值的均值分布。CLTV大约为75， DTI大约为30，LTV则接近80.

备注

[1] CLTV: 原始组合贷款价值

- 在购买抵押贷款的情况下，该比率是通过将票据日期的原始抵押贷款金额加上卖方披露的任何二级抵押贷款金额除以抵押财产在票据日或其购买价格上的评估价值。在再融资抵押贷款的情况下，该比率是通过将票据日期的原始抵押贷款金额加上卖方披露的任何二级抵押贷款金额除以抵押日期的抵押房地产的评估价值获得的。如果卖方披露的二次融资金额包括房屋净值信贷额度，那么CLTV计算反映了第一次留置权抵押贷款结束时的已支付金额，而不是房屋净值信贷额度下可用的最高贷款金额。对于经验丰富的抵押贷款，如果卖方不能保证抵押财产的价值自注释日期以来没有下降，则Freddie Mac要求卖方必须提供新的评估值，用于CLTV计算。在某些情况下，如果卖方向Freddie Mac提供贷款，并附有指示额外二级抵押贷款金额的特殊代码，则这些金额可能已包含在CLTV计算中。

[2] DTI:原始债务收入比率

- 债务与收入比率的披露基于（1）借款人每月债务支付的总和，包括纳入借款人在当时支付的抵押支付的月度住房支出。 将抵押贷款交付给房地美，除以（2）用于在该贷款发起之日承保贷款的每月总收入。

[3] LTV: 原始贷款价值

- 在购买抵押贷款的情况下，通过将票据日期的原始抵押贷款金额除以抵押日期或其购买价格中抵押房产的评估价值中的较小者而获得的比率。在再融资抵押贷款的情况下，通过将票据日期的原始抵押贷款金额与抵押财产在评估日的评估价值除以获得的比率。对于经验丰富的抵押贷款，如果卖方不能保证抵押财产的价值自注明日期以来没有下降，则Freddie Mac要求卖方必须提供新的评估值，用于LTV计算。