In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv(r"C:\Users\hp\Downloads\Loan_Default.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	ID	year	loan_limit	Gender	approv_in_adv	loan_type	loan_purpose	Credit_Worthiness
0	24890	2019	cf	Sex Not Available	nopre	type1	p1	l1
1	24891	2019	cf	Male	nopre	type2	p1	l1
2	24892	2019	cf	Male	pre	type1	p1	l1
3	24893	2019	cf	Male	nopre	type1	p4	l1
4	24894	2019	cf	Joint	pre	type1	p1	l1

5 rows × 34 columns

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

Out[4]:
```
(148670, 34)
```

Missing Value Treatment

In [5]:
```
df.isnull().sum()  # count of how many null/missing values are present in each colu
```

Out[5]:

```
ID                          0
year                        0
loan_limit               3344
Gender                      0
approv_in_adv             908
loan_type                   0
loan_purpose              134
Credit_Worthiness           0
open_credit                 0
business_or_commercial      0
loan_amount                 0
rate_of_interest        36439
Interest_rate_spread    36639
Upfront_charges         39642
term                       41
Neg_ammortization         121
interest_only               0
lump_sum_payment            0
property_value          15098
construction_type           0
occupancy_type              0
Secured_by                  0
total_units                 0
income                   9150
credit_type                 0
Credit_Score                0
co-applicant_credit_type    0
age                       200
submission_of_application 200
LTV                     15098
Region                      0
Security_Type               0
Status                      0
dtir1                   24121
dtype: int64
```

In [6]:

```
df.isnull().sum()*100/len(df)
```

Out[6]:
```
ID                          0.000000
year                        0.000000
loan_limit                  2.249277
Gender                      0.000000
approv_in_adv               0.610749
loan_type                   0.000000
loan_purpose                0.090133
Credit_Worthiness           0.000000
open_credit                 0.000000
business_or_commercial      0.000000
loan_amount                 0.000000
rate_of_interest           24.509989
Interest_rate_spread       24.644515
Upfront_charges            26.664425
term                        0.027578
Neg_ammortization           0.081388
interest_only               0.000000
lump_sum_payment            0.000000
property_value             10.155378
construction_type           0.000000
occupancy_type              0.000000
Secured_by                  0.000000
total_units                 0.000000
income                      6.154571
credit_type                 0.000000
Credit_Score                0.000000
co-applicant_credit_type    0.000000
age                         0.134526
submission_of_application   0.134526
LTV                        10.155378
Region                      0.000000
Security_Type               0.000000
Status                      0.000000
dtir1                      16.224524
dtype: float64
```

In [7]:
```
def check_missing(x):
    x=x.isnull().sum()*100/len(x)
    x=x[x>0]
    x=x.sort_values(ascending=False)
    return x

check_missing(df)
```

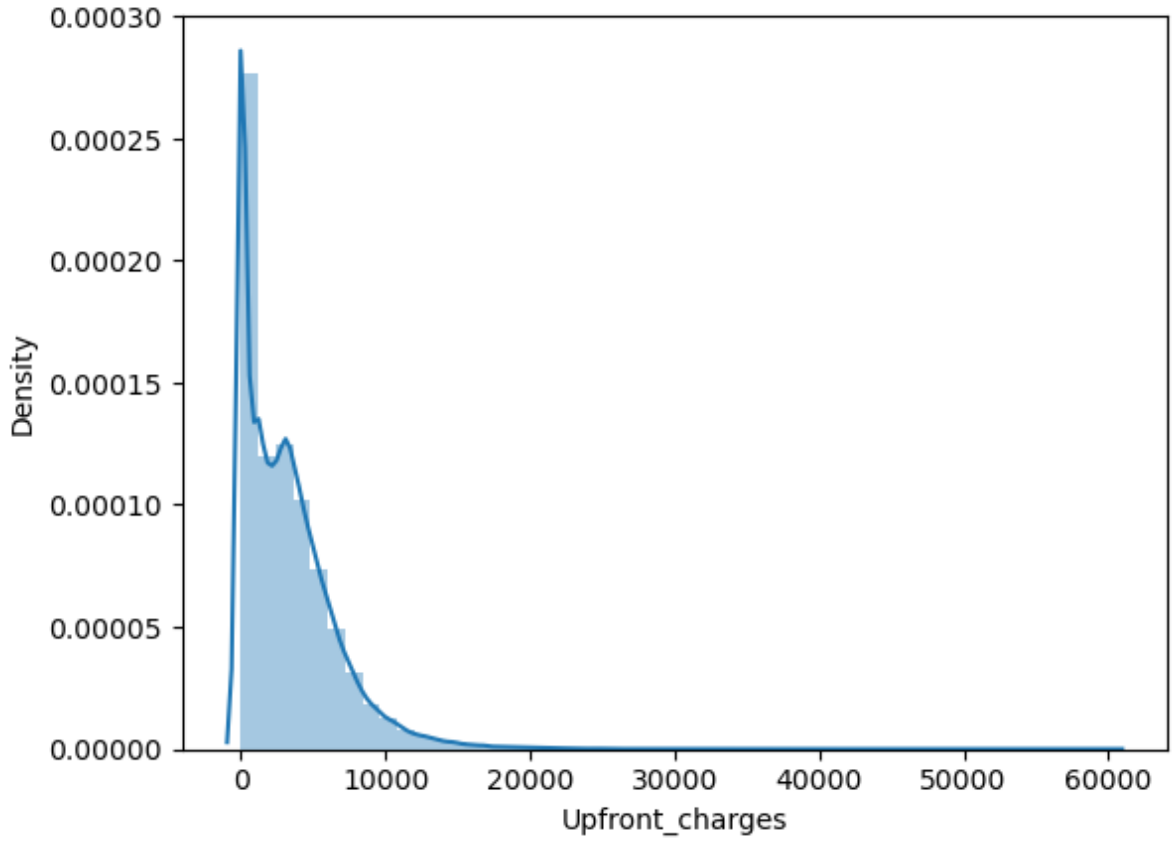
Out[7]:
```
Upfront_charges            26.664425
Interest_rate_spread       24.644515
rate_of_interest           24.509989
dtir1                      16.224524
property_value             10.155378
LTV                        10.155378
income                      6.154571
loan_limit                  2.249277
approv_in_adv               0.610749
age                         0.134526
submission_of_application   0.134526
loan_purpose                0.090133
Neg_ammortization           0.081388
term                        0.027578
dtype: float64
```

In [8]:
```
df.Upfront_charges[:10]
```

Out[8]:
```
0        NaN
1        NaN
2     595.00
3        NaN
4       0.00
5     370.00
6    5120.00
7    5609.88
8    1150.00
9    2316.50
Name: Upfront_charges, dtype: float64
```

In [9]:
```
sns.distplot(df['Upfront_charges'])
```

Out[9]:
```
<Axes: xlabel='Upfront_charges', ylabel='Density'>
```



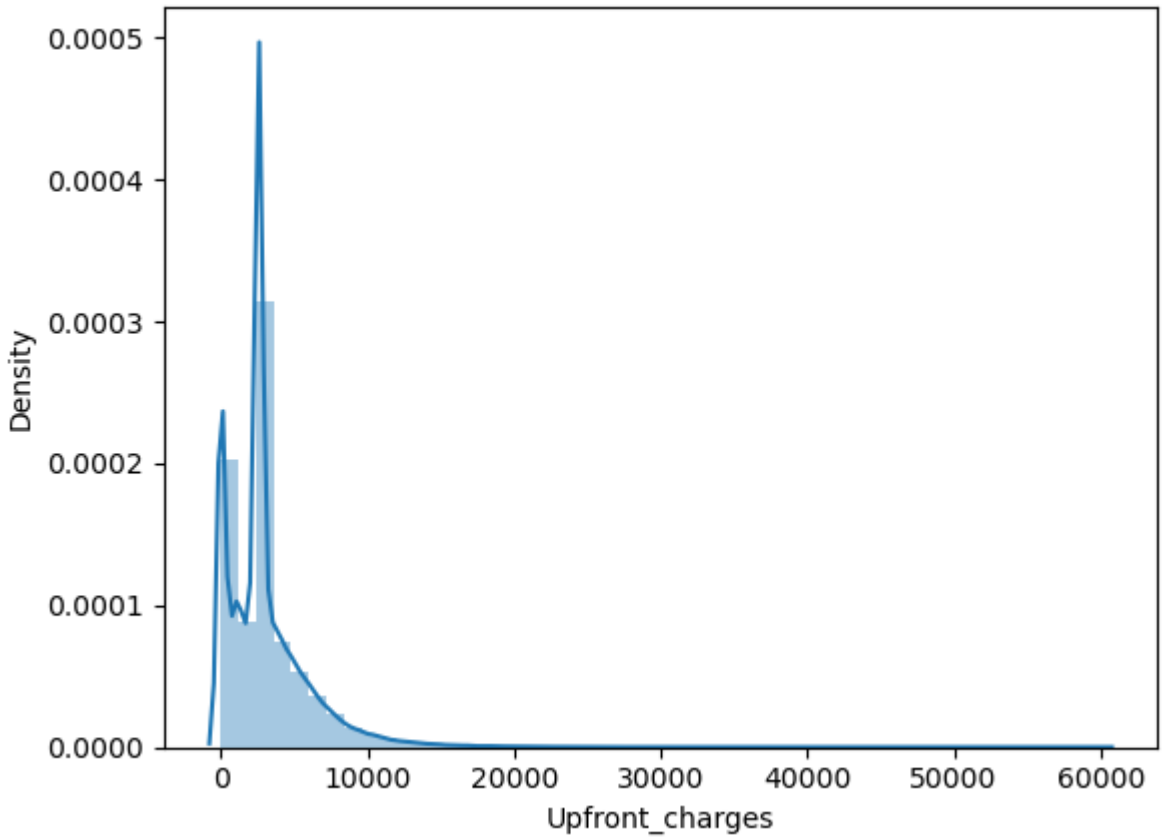
In [10]:
```
df['Upfront_charges'].median()
```

Out[10]:
```
2596.45
```

In [11]:
```
df.Upfront_charges = df.Upfront_charges.fillna(2596.45)
```

In [12]:
```
sns.distplot(df['Upfront_charges'])
```

Out[12]:
```
<Axes: xlabel='Upfront_charges', ylabel='Density'>
```



In [13]:
```
check_missing(df)
```

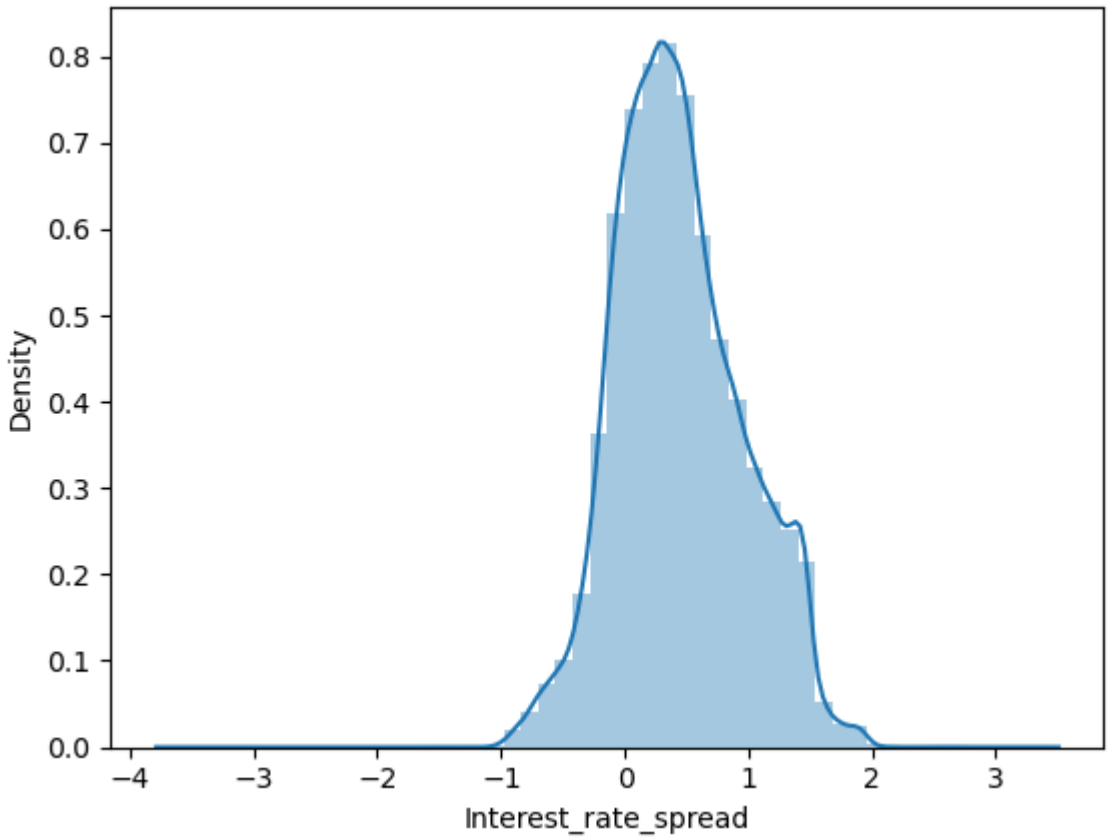
Out[13]:
```
Interest_rate_spread         24.644515
rate_of_interest             24.509989
dtir1                        16.224524
property_value               10.155378
LTV                          10.155378
income                        6.154571
loan_limit                    2.249277
approv_in_adv                 0.610749
age                           0.134526
submission_of_application     0.134526
loan_purpose                  0.090133
Neg_ammortization             0.081388
term                          0.027578
dtype: float64
```

In [14]:
```
df.Interest_rate_spread
```

Out[14]:
```
0            NaN
1            NaN
2         0.2000
3         0.6810
4         0.3042
           ...
148665    0.2571
148666    0.8544
148667    0.0816
148668    0.5824
148669    1.3871
Name: Interest_rate_spread, Length: 148670, dtype: float64
```

In [15]:
```
sns.distplot(df.Interest_rate_spread)
```

Out[15]:
```
<Axes: xlabel='Interest_rate_spread', ylabel='Density'>
```



In [16]:
```
df.Interest_rate_spread.mean()
```

Out[16]:
```
0.4416556604868295
```

In [17]:
```
df.Interest_rate_spread.median()
```

Out[17]:
```
0.3904
```

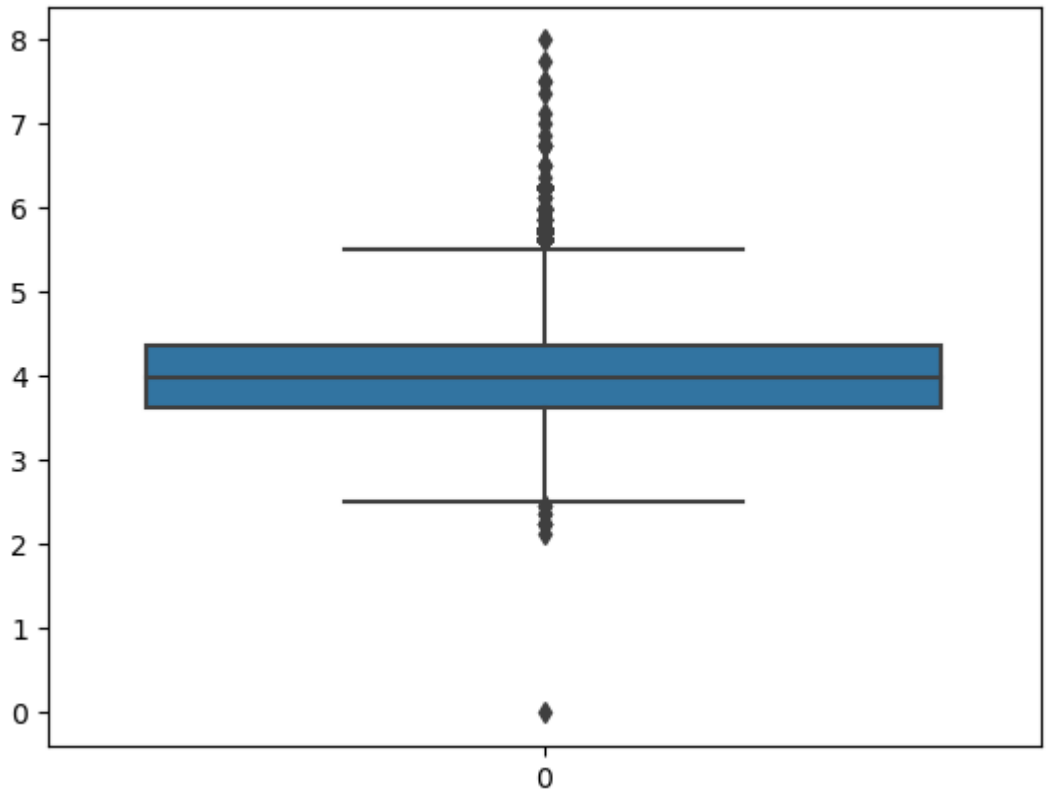
In [18]:
```
df.Interest_rate_spread = df.Interest_rate_spread.fillna(df.Interest_rate_spread.me
```

In [19]:
```
check_missing(df)
```

Out[19]:
```
rate_of_interest             24.509989
dtir1                        16.224524
property_value               10.155378
LTV                          10.155378
income                        6.154571
loan_limit                    2.249277
approv_in_adv                 0.610749
age                           0.134526
submission_of_application     0.134526
loan_purpose                  0.090133
Neg_ammortization             0.081388
term                          0.027578
dtype: float64
```

In [20]:
```
sns.boxplot(df.rate_of_interest)
```

Out[20]:
```
<Axes: >
```



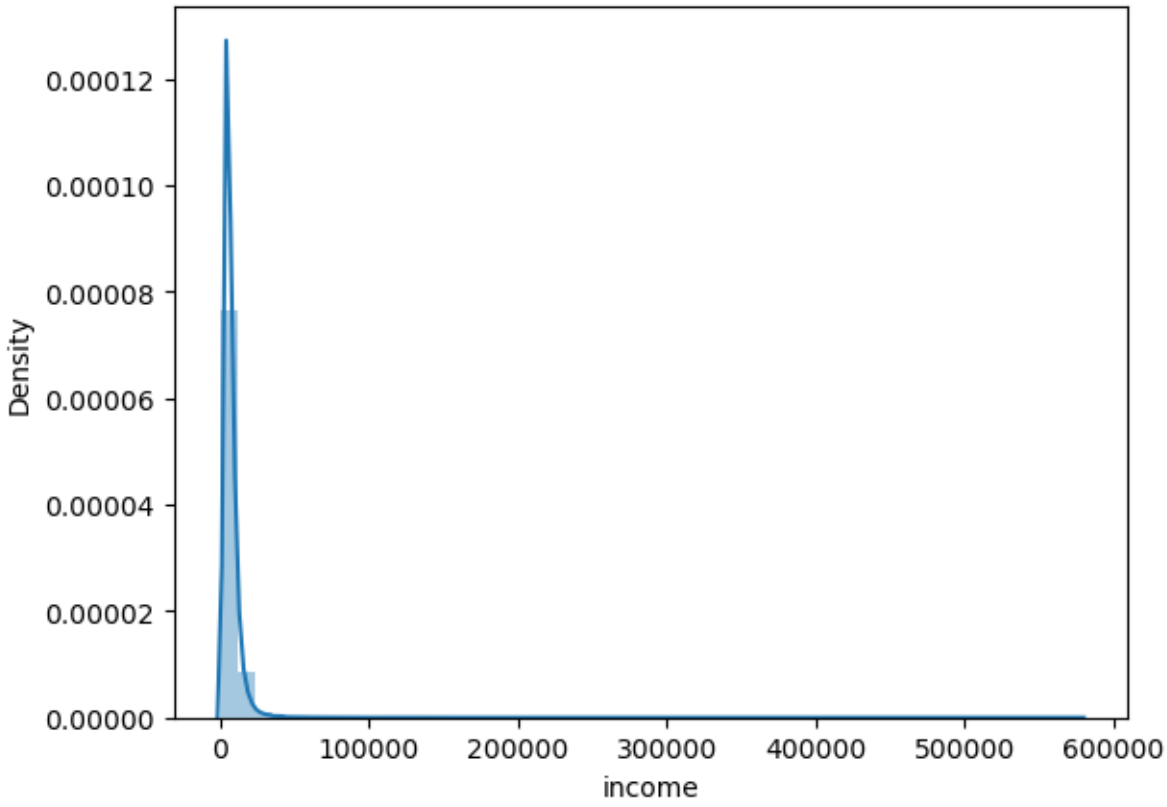
In [21]:
```
df.rate_of_interest =df.rate_of_interest.fillna(df.rate_of_interest.mean())
```

In [22]:
```
check_missing(df)
```

Out[22]:
```
dtir1                       16.224524
property_value              10.155378
LTV                         10.155378
income                       6.154571
loan_limit                   2.249277
approv_in_adv                0.610749
age                          0.134526
submission_of_application    0.134526
loan_purpose                 0.090133
Neg_ammortization            0.081388
term                         0.027578
dtype: float64
```

In [23]:
```
sns.distplot(df.income)
```

Out[23]:
```
<Axes: xlabel='income', ylabel='Density'>
```



In [24]:
```
df.income = df.income.fillna(df.income.mean())
```

In [25]:
```
check_missing(df)
```

Out[25]:
```
dtir1                        16.224524
property_value               10.155378
LTV                          10.155378
loan_limit                    2.249277
approv_in_adv                 0.610749
age                           0.134526
submission_of_application     0.134526
loan_purpose                  0.090133
Neg_ammortization             0.081388
term                          0.027578
dtype: float64
```

Check for data type errors

In [26]:
```
df.dtypes
```

Out[26]:
```
ID                          int64
year                        int64
loan_limit                 object
Gender                     object
approv_in_adv              object
loan_type                  object
loan_purpose               object
Credit_Worthiness          object
open_credit                object
business_or_commercial     object
loan_amount                 int64
rate_of_interest           float64
Interest_rate_spread       float64
Upfront_charges            float64
term                       float64
Neg_ammortization          object
interest_only              object
lump_sum_payment           object
property_value             float64
construction_type          object
occupancy_type             object
Secured_by                 object
total_units                object
income                     float64
credit_type                object
Credit_Score                int64
co-applicant_credit_type   object
age                        object
submission_of_application  object
LTV                        float64
Region                     object
Security_Type              object
Status                      int64
dtir1                      float64
dtype: object
```

In [27]:
```
df.Neg_ammortization.unique()
```

Out[27]:
```
array(['not_neg', 'neg_amm', nan], dtype=object)
```

In [28]:
```
for col in df.select_dtypes('O').columns:
    print(col)
    print(df[col].unique()[:10])
```

```
loan_limit
['cf' nan 'ncf']
Gender
['Sex Not Available' 'Male' 'Joint' 'Female']
approv_in_adv
['nopre' 'pre' nan]
loan_type
['type1' 'type2' 'type3']
loan_purpose
['p1' 'p4' 'p3' 'p2' nan]
Credit_Worthiness
['l1' 'l2']
open_credit
['nopc' 'opc']
business_or_commercial
['nob/c' 'b/c']
Neg_ammortization
['not_neg' 'neg_amm' nan]
interest_only
['not_int' 'int_only']
lump_sum_payment
['not_lpsm' 'lpsm']
construction_type
['sb' 'mh']
occupancy_type
['pr' 'sr' 'ir']
Secured_by
['home' 'land']
total_units
['1U' '2U' '3U' '4U']
credit_type
['EXP' 'EQUI' 'CRIF' 'CIB']
co-applicant_credit_type
['CIB' 'EXP']
age
['25-34' '55-64' '35-44' '45-54' '65-74' '>74' '<25' nan]
submission_of_application
['to_inst' 'not_inst' nan]
Region
['south' 'North' 'central' 'North-East']
Security_Type
['direct' 'Indriect']
```

Explorartory Data Analysis

In [29]:
```
### Univariate Analysis -> Where we try to understand behaviour of 1 feature/attrib
### Bivariate Analysis -> We try to understand relationship between 2 or more attri
### Type of Charts

## line chart -> To Understand Trend or time series Data or evolution of data with
## Bar chart -> To understand distribution across categorical Features
## Histogram -> To understand distribution of Numeric Fetaure
## Boxplot
## Pie
## Dist Plot
## KDE Plots
## Scatter Plot ****
## Heat Maps
## Contour Plots
## Voilin Plots
## Bee Swarm Plots
```

https://i.redd.it/3z3ap9j8cj551.png

Target Variable -> Status

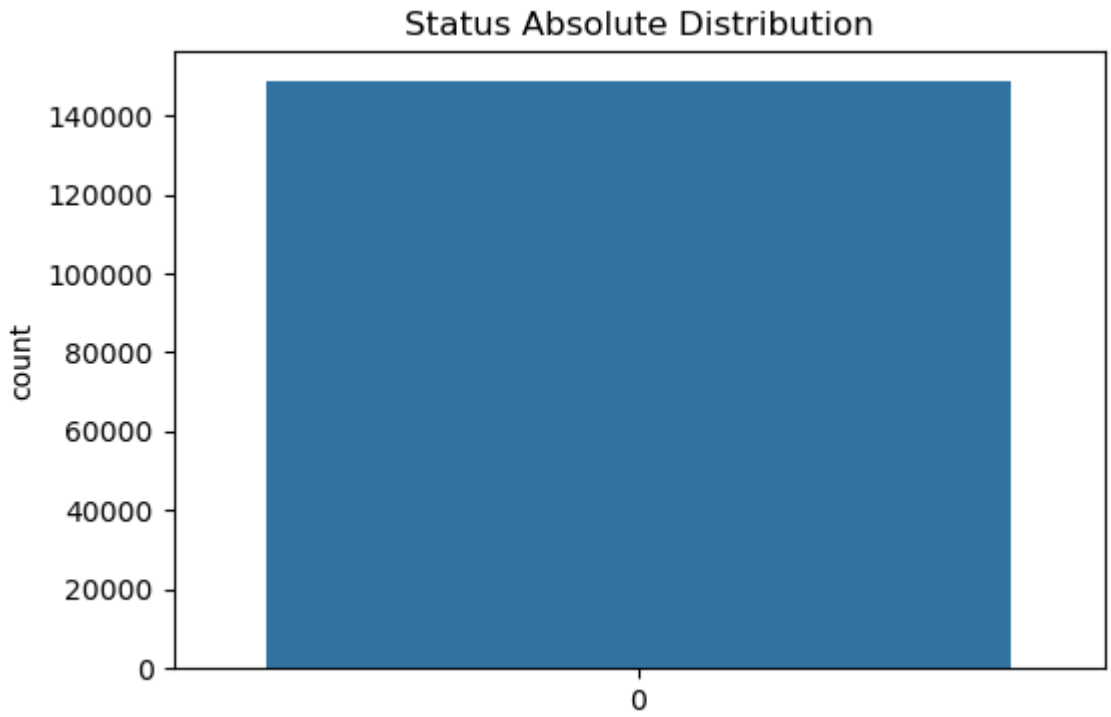
In [30]:
```
df['Status'].value_counts()
```

Out[30]:
```
Status
0    112031
1     36639
Name: count, dtype: int64
```

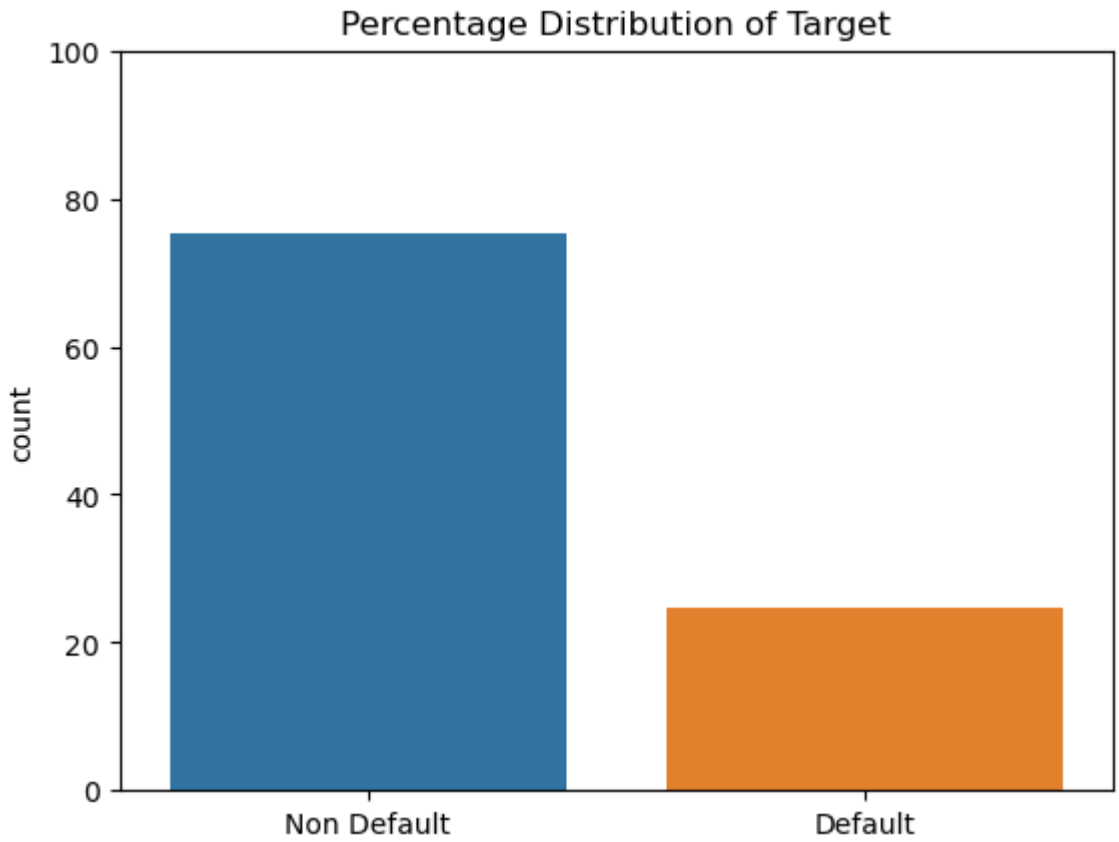
In [31]:
```
df['Status'].value_counts()*100/len(df)
```

Out[31]:
```
Status
0    75.355485
1    24.644515
Name: count, dtype: float64
```

In [32]:
```
plt.figure(figsize=(6,4),dpi=100)
sns.countplot(df.Status)
plt.title('Status Absolute Distribution')
plt.show()
```



In [33]:
```
temp = df['Status'].value_counts()*100/len(df)
plt.figure()
sns.barplot(x = ['Non Default','Default'],y=temp)
plt.ylim(0,100)
plt.title('Percentage Distribution of Target')
plt.show()
```



The current data showcase that we have 75% population as non-defaulters while 25% Population have defaulted

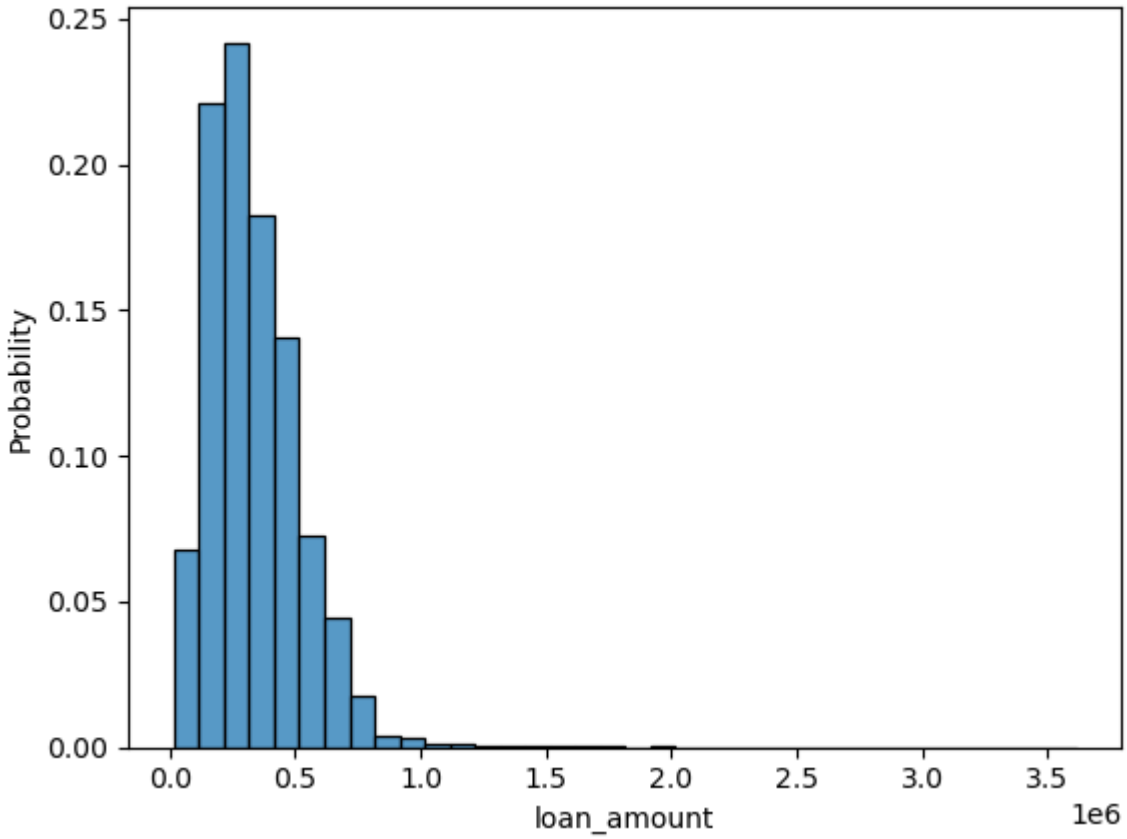
In [34]:
```
num_cols = [col for col in df.columns if col not in df.select_dtypes('O').columns]
num_cols
```

Out[34]:
```
['ID',
 'year',
 'loan_amount',
 'rate_of_interest',
 'Interest_rate_spread',
 'Upfront_charges',
 'term',
 'property_value',
 'income',
 'Credit_Score',
 'LTV',
 'Status',
 'dtir1']
```

Loan Amount

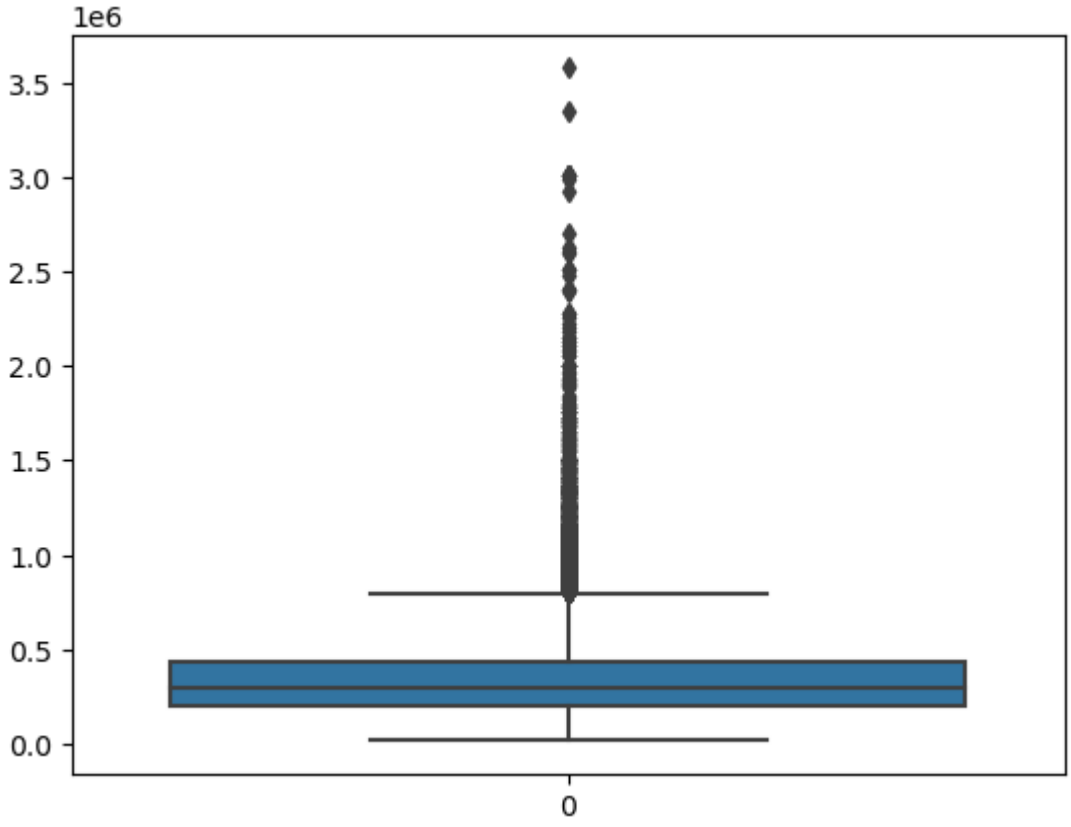
In [35]:
```
sns.histplot(df.loan_amount,binwidth=100000,stat='probability')
```

Out[35]:
```
<Axes: xlabel='loan_amount', ylabel='Probability'>
```



In [36]:
```
sns.boxplot(df.loan_amount)
```

Out[36]:
```
<Axes: >
```



In [37]:
```
len(df[df.loan_amount<=500000])/len(df)
```

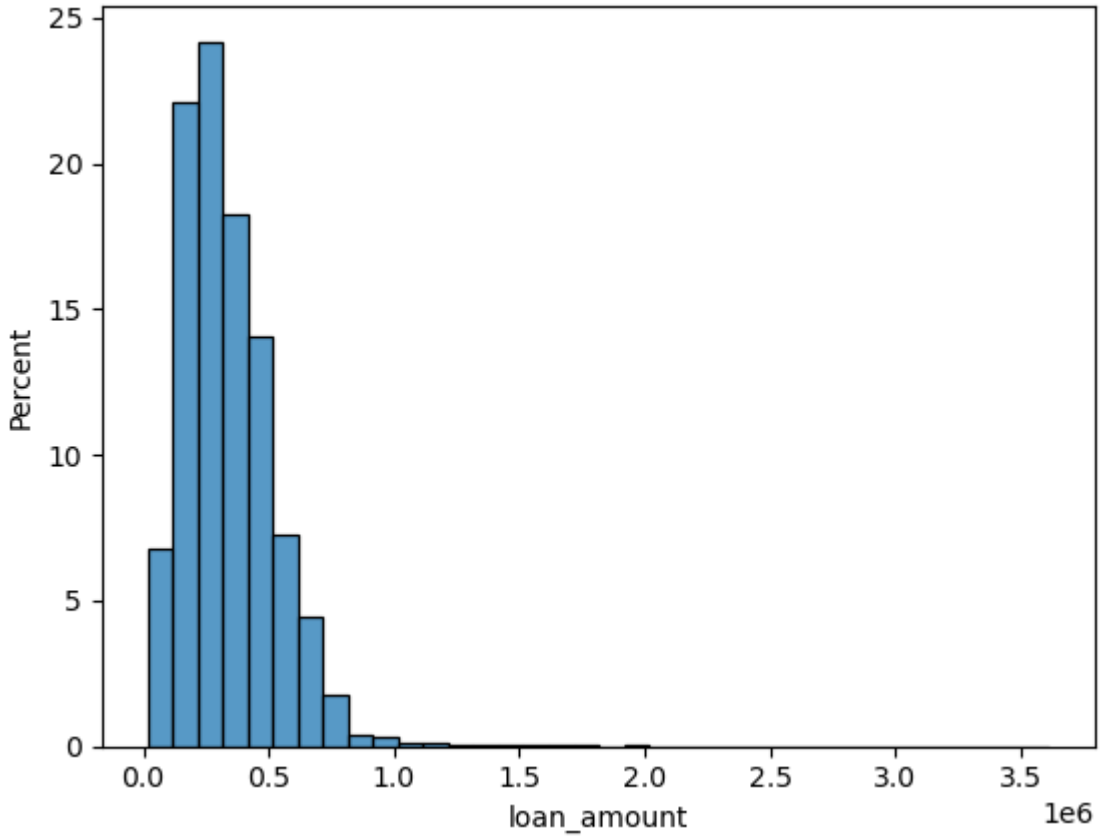
Out[37]:
```
0.8452478643976592
```

In [38]:
```
len(df[(df.loan_amount<=500000) &  (df.loan_amount >=100000)])/len(df)
```

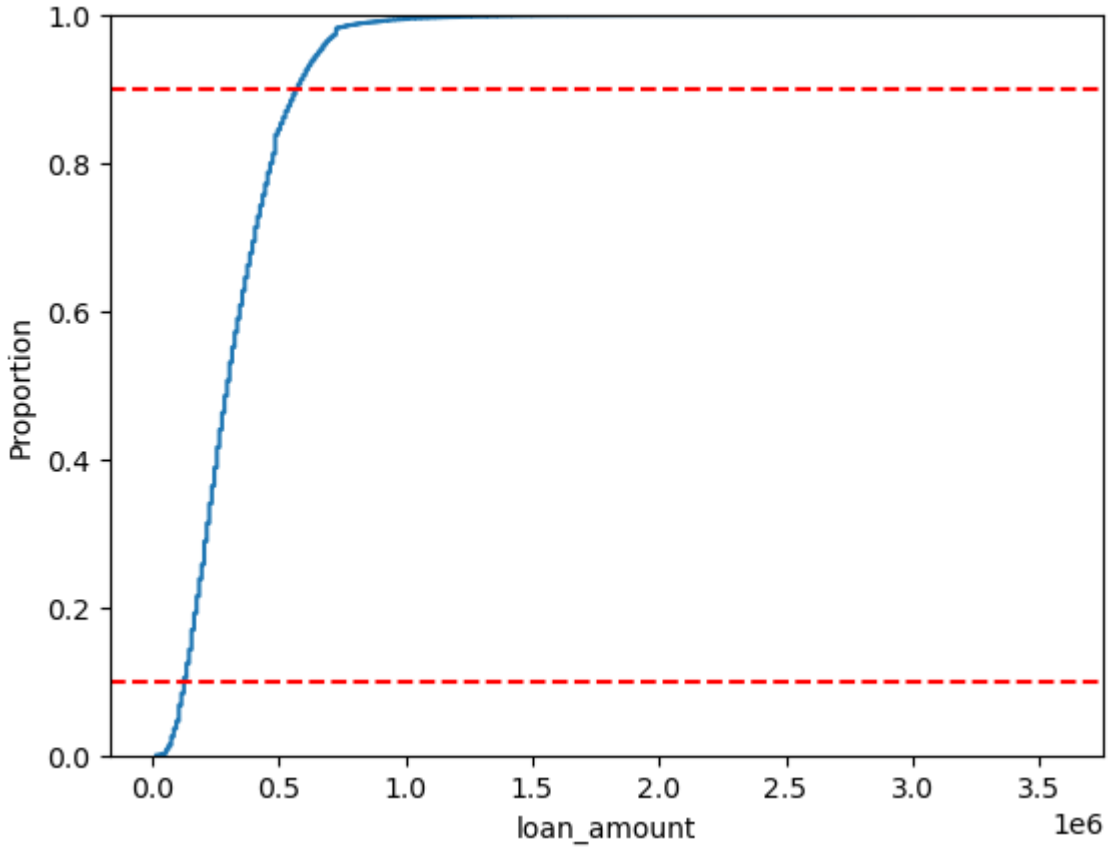
Out[38]:
```
0.79937445348759
```

In [39]:
```
sns.histplot(df.loan_amount,binwidth=100000,stat='percent')
```

Out[39]:
```
<Axes: xlabel='loan_amount', ylabel='Percent'>
```



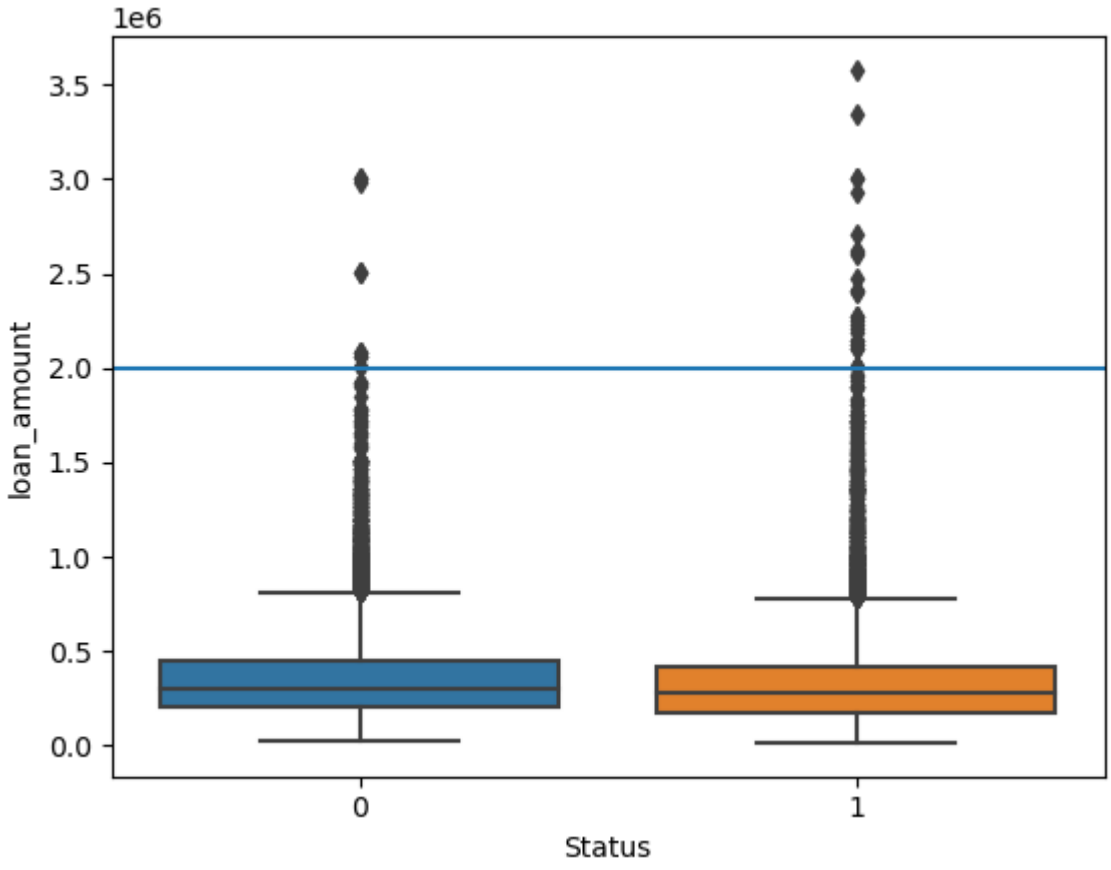
In [40]:
```
sns.ecdfplot(df.loan_amount)
plt.axhline(0.1,color='red',linestyle='--')
plt.axhline(0.9,color='red',linestyle='--')
plt.show()
```



In [41]:
```
sns.ecdfplot?
```

In [42]:
```
sns.boxplot(x='Status',y='loan_amount',data = df)
plt.axhline(2000000)
```

Out[42]:
```
<matplotlib.lines.Line2D at 0x19e15d19290>
```



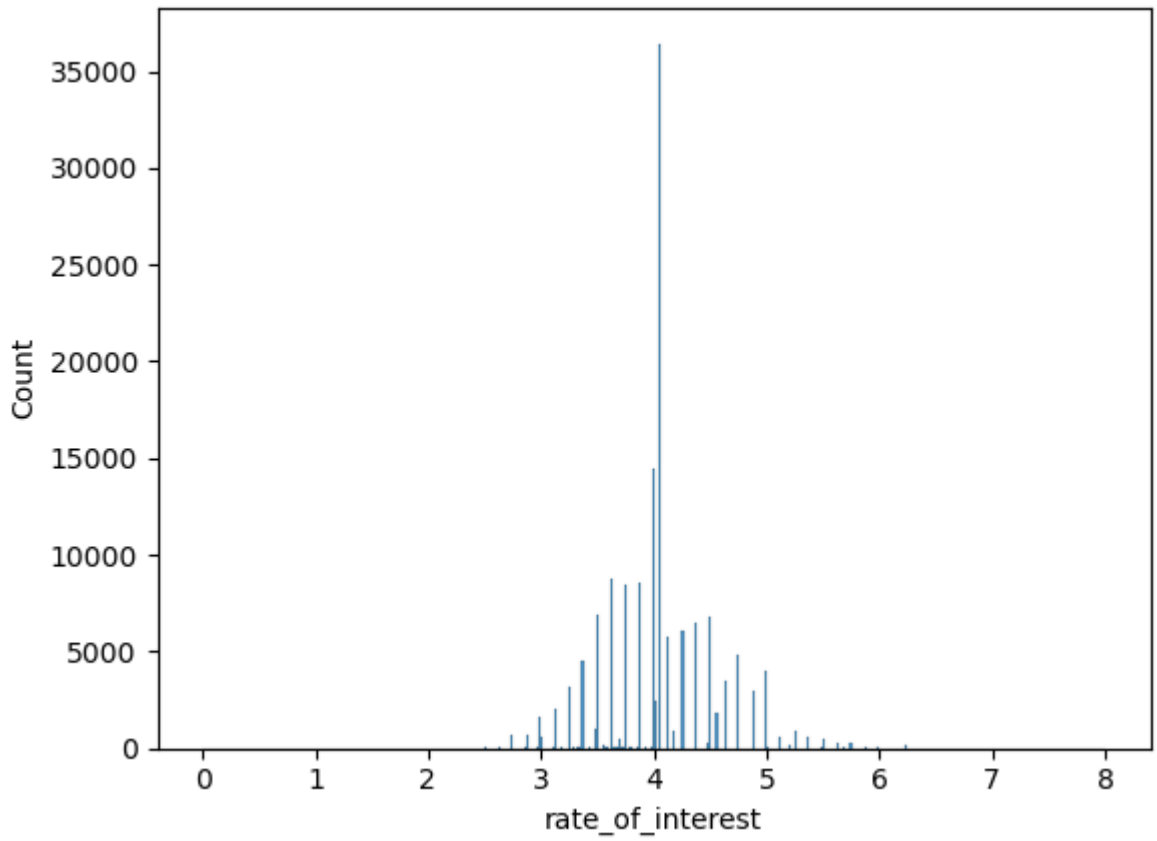
80% of popultion have taken loan between 100,000 and 600,000

Probability of Default in Loan Increases above 2M loan amount

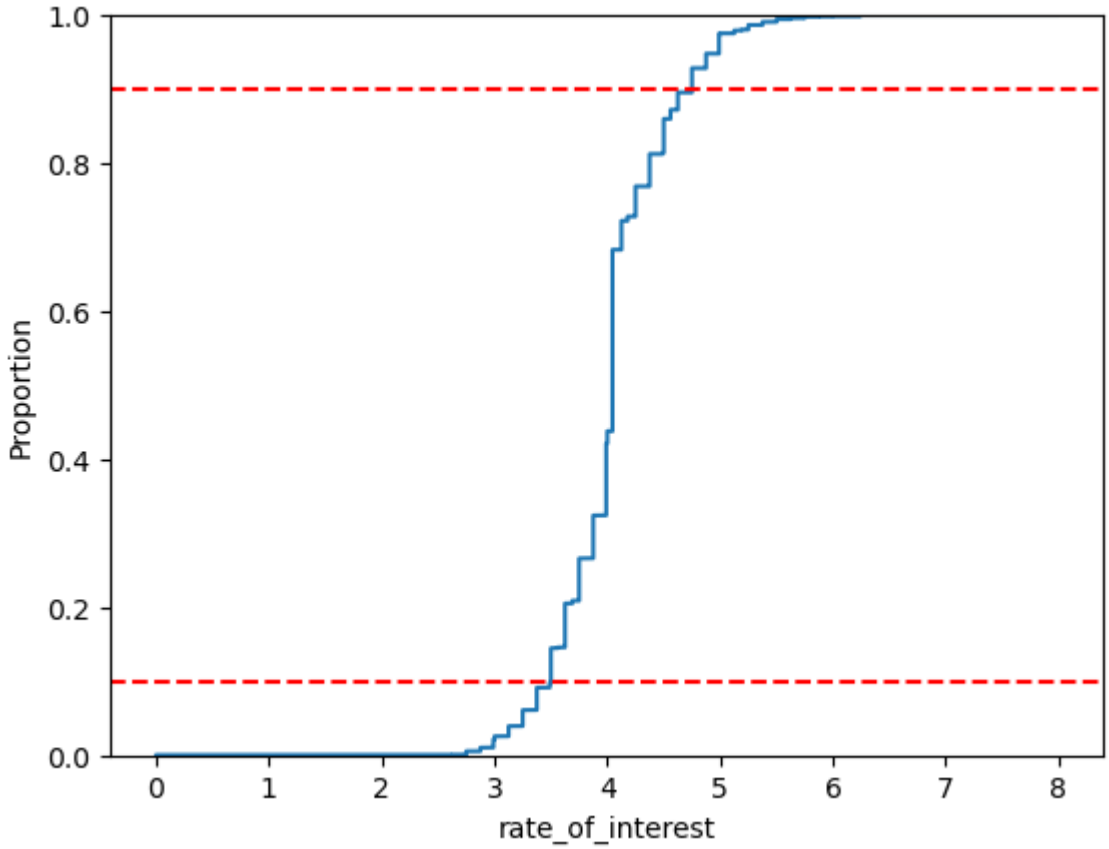
Rate_Of_Interest

In [43]:
```
sns.histplot(df.rate_of_interest)
```

Out[43]:
```
<Axes: xlabel='rate_of_interest', ylabel='Count'>
```

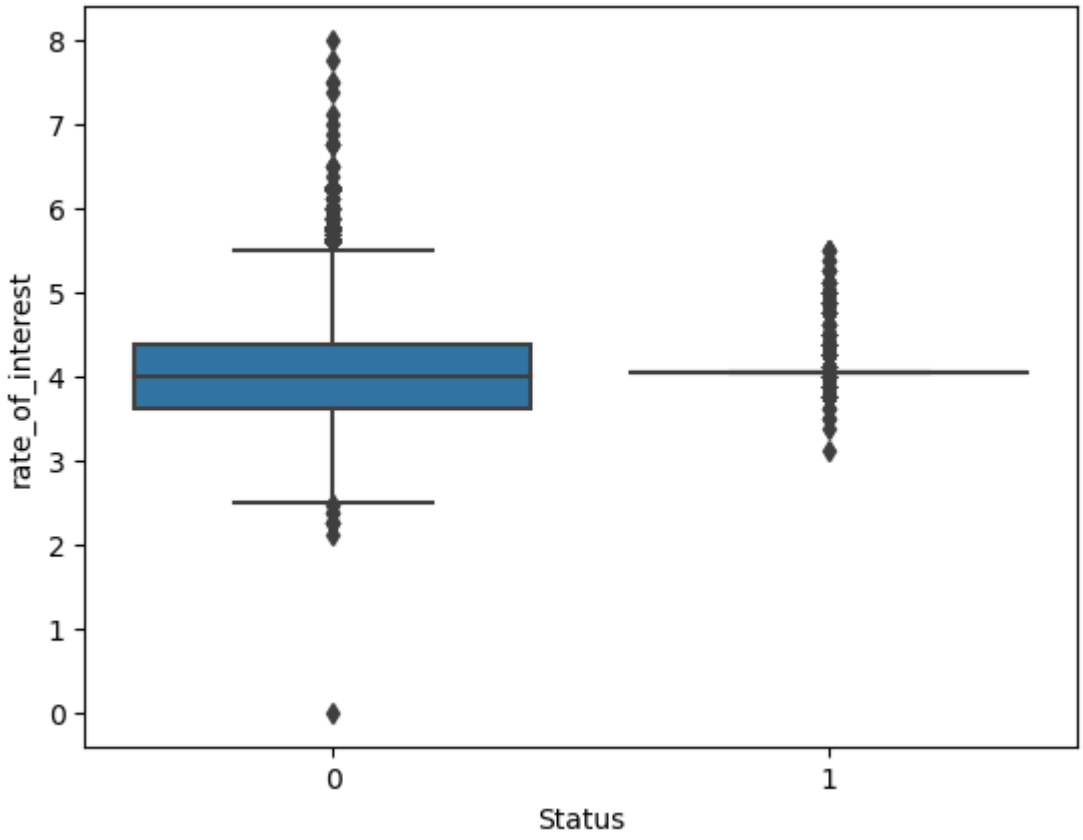


In [44]:
```
sns.ecdfplot(df.rate_of_interest)
plt.axhline(0.1,color='red',linestyle='--')
plt.axhline(0.9,color='red',linestyle='--')
plt.show()
```



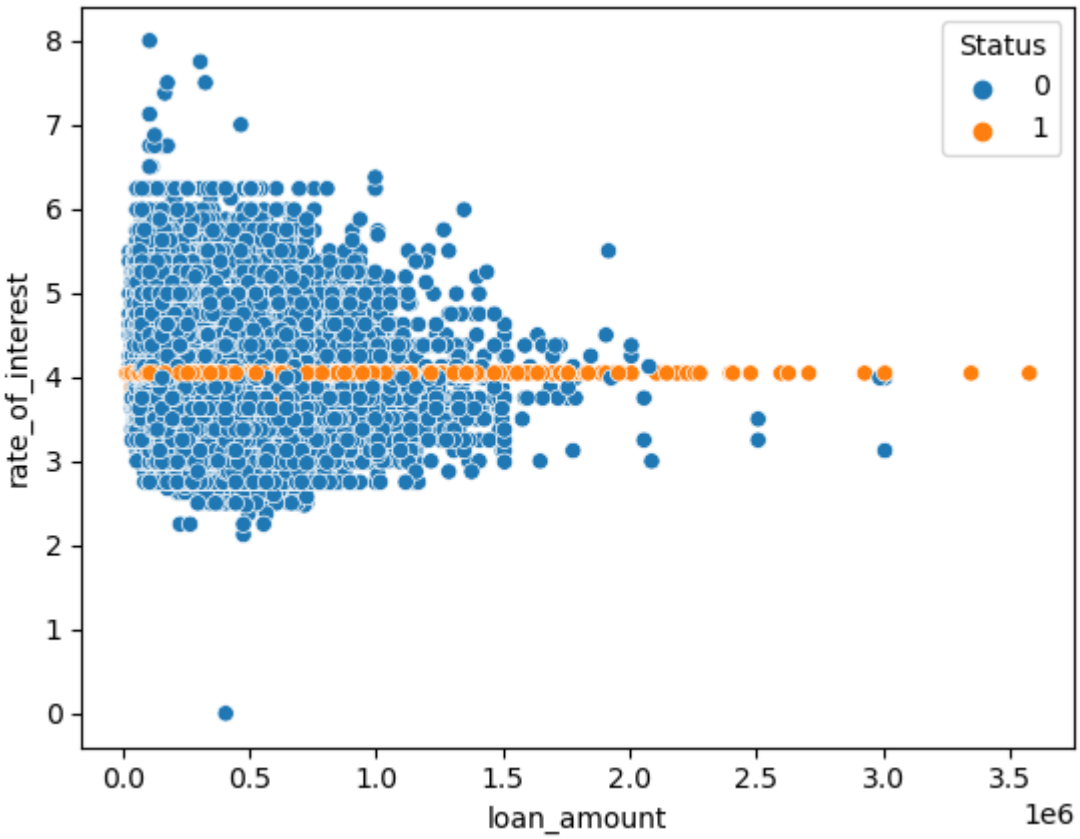
In [45]:
```
sns.boxplot(x='Status',y='rate_of_interest',data = df)
```

Out[45]:
```
<Axes: xlabel='Status', ylabel='rate_of_interest'>
```



In [46]:
```
sns.scatterplot(x='loan_amount',y='rate_of_interest',data=df,hue='Status')
```

Out[46]:
```
<Axes: xlabel='loan_amount', ylabel='rate_of_interest'>
```

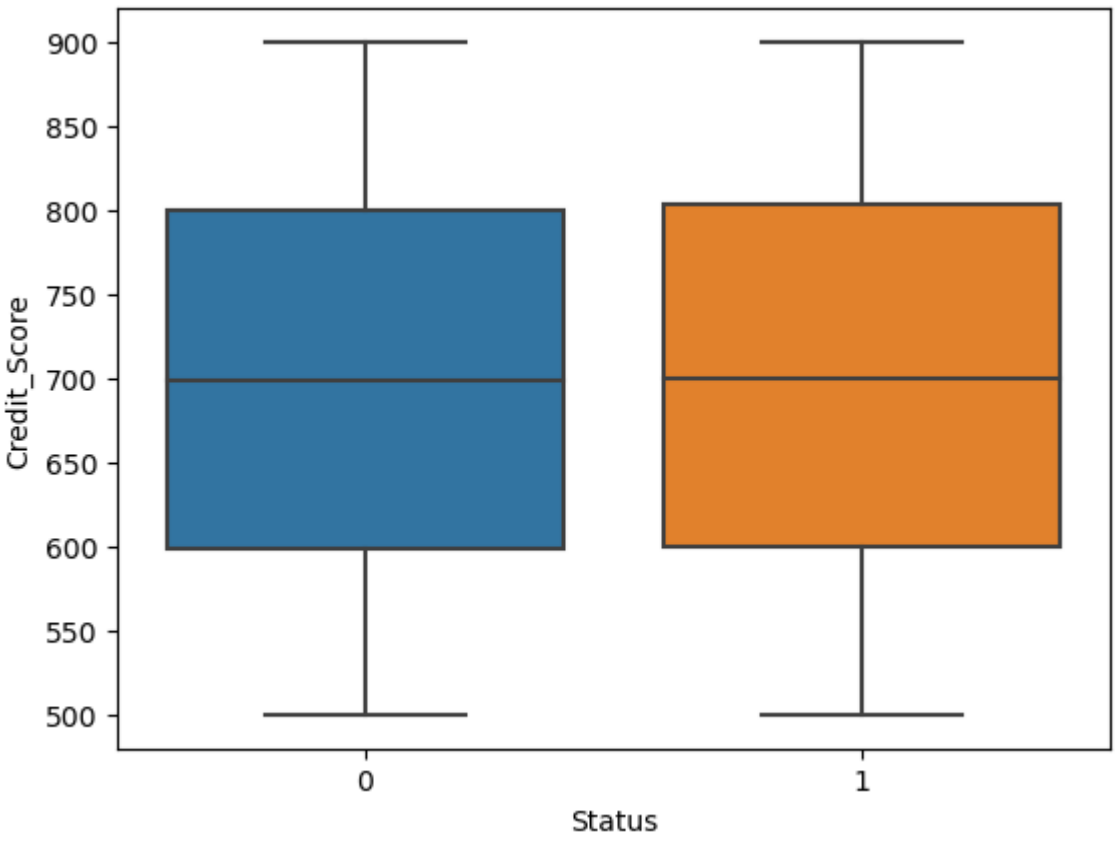


Rate of Intereset doesn't have a prominent impact on default rate

Credit_Score

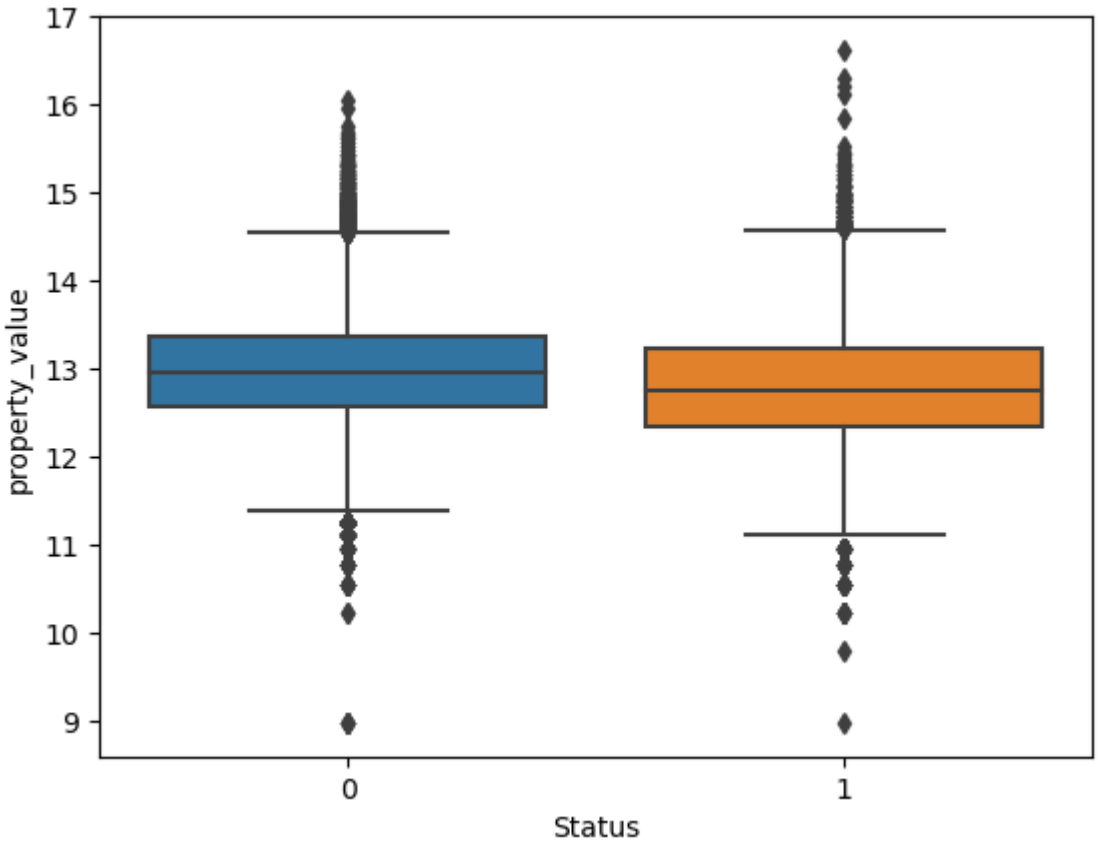
In [47]:
```
sns.boxplot(x='Status',y='Credit_Score',data = df)
```

Out[47]:
```
<Axes: xlabel='Status', ylabel='Credit_Score'>
```



In [48]:
```
sns.boxplot(x=df.Status,y=df.property_value.apply(np.log),data=df)
```

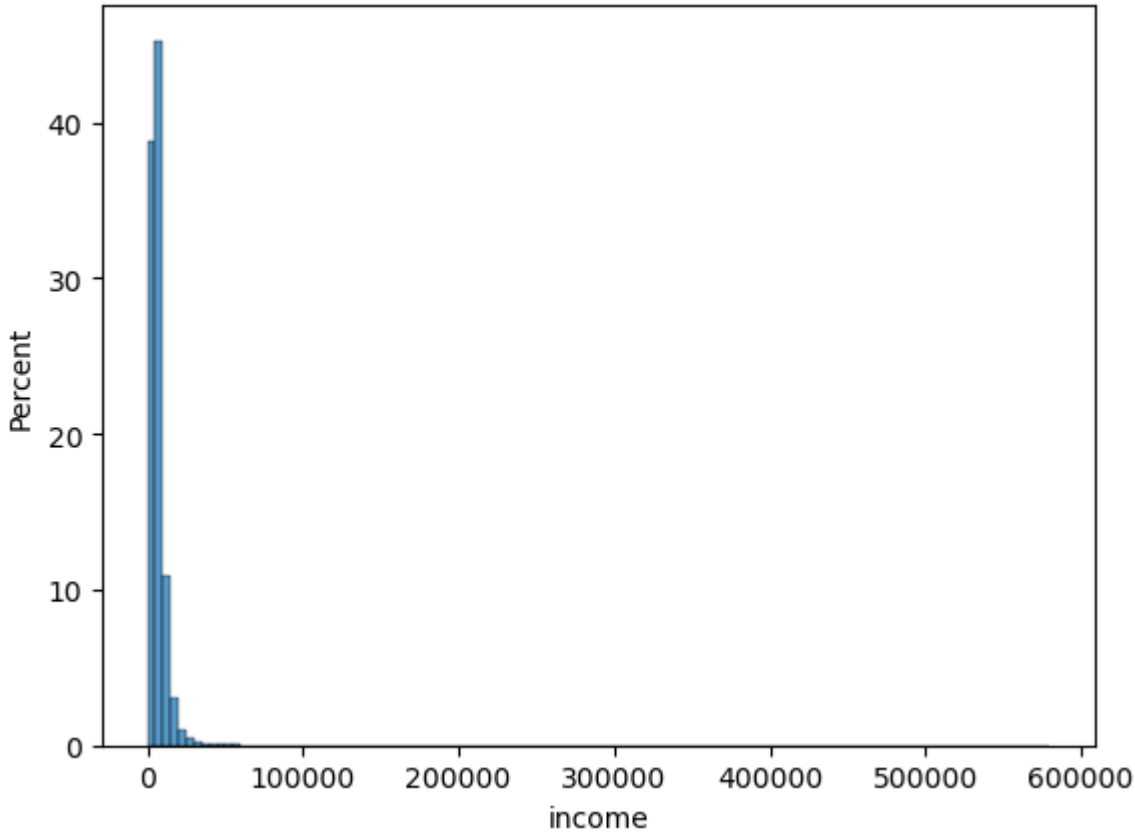
Out[48]:
```
<Axes: xlabel='Status', ylabel='property_value'>
```



income

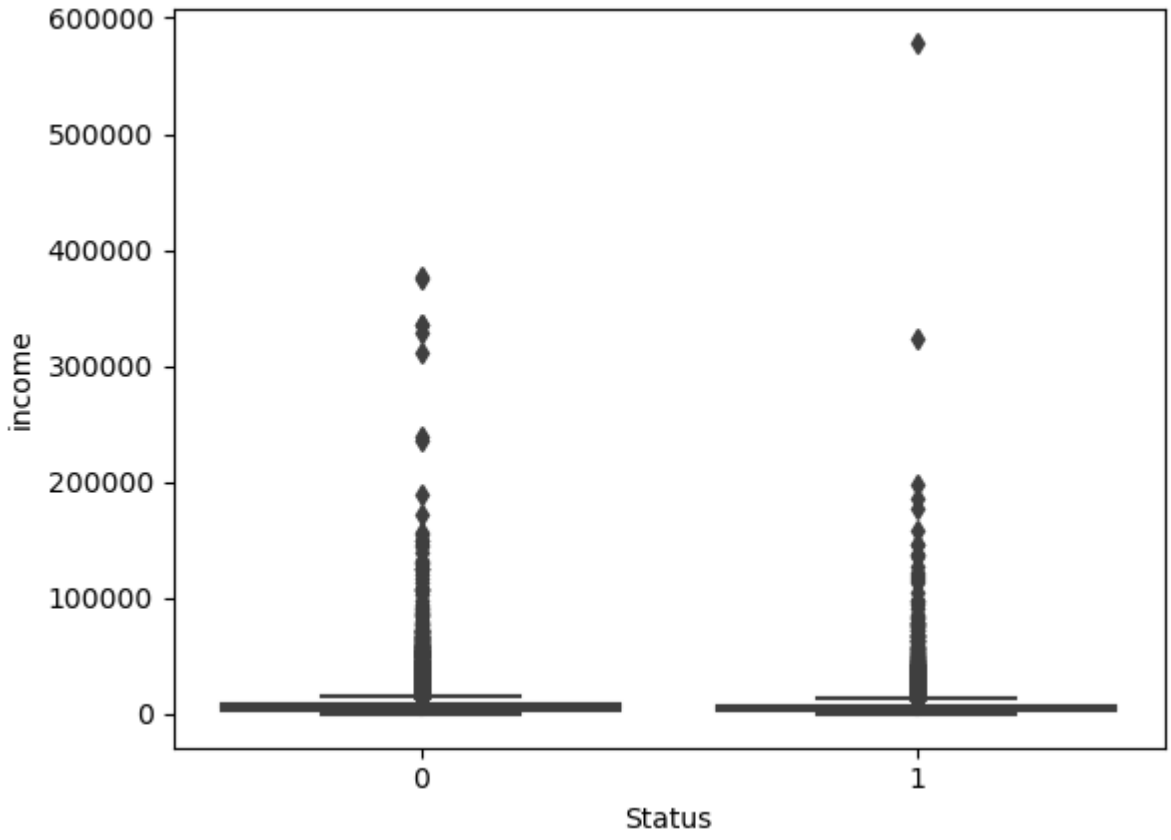
In [49]:
```
sns.histplot(df.income,bins=100,binwidth=5000,stat='percent')
```

Out[49]:
```
<Axes: xlabel='income', ylabel='Percent'>
```



In [50]:
```
sns.boxplot(x='Status',y='income',data = df)
```

Out[50]:
```
<Axes: xlabel='Status', ylabel='income'>
```



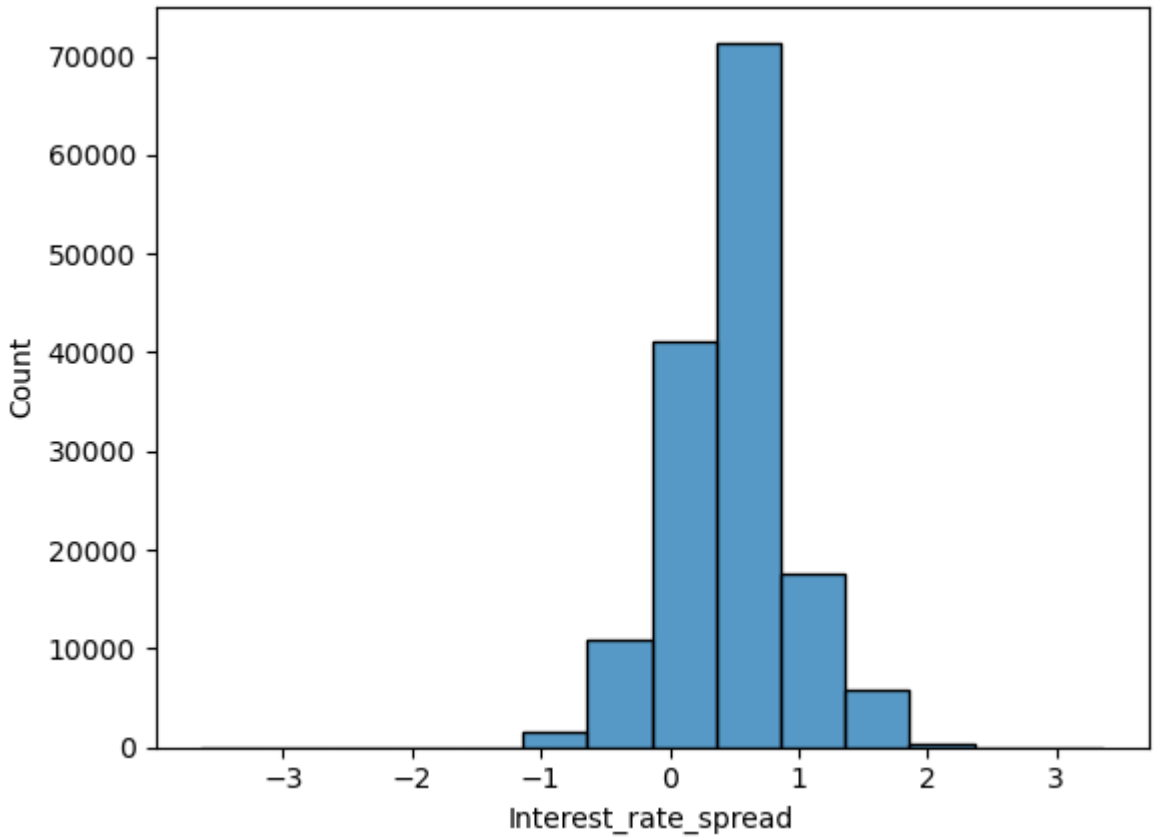
Who has more salaries are having less chance of default.

Who has having less salaries are having move default rate.

Interest_rate_spread

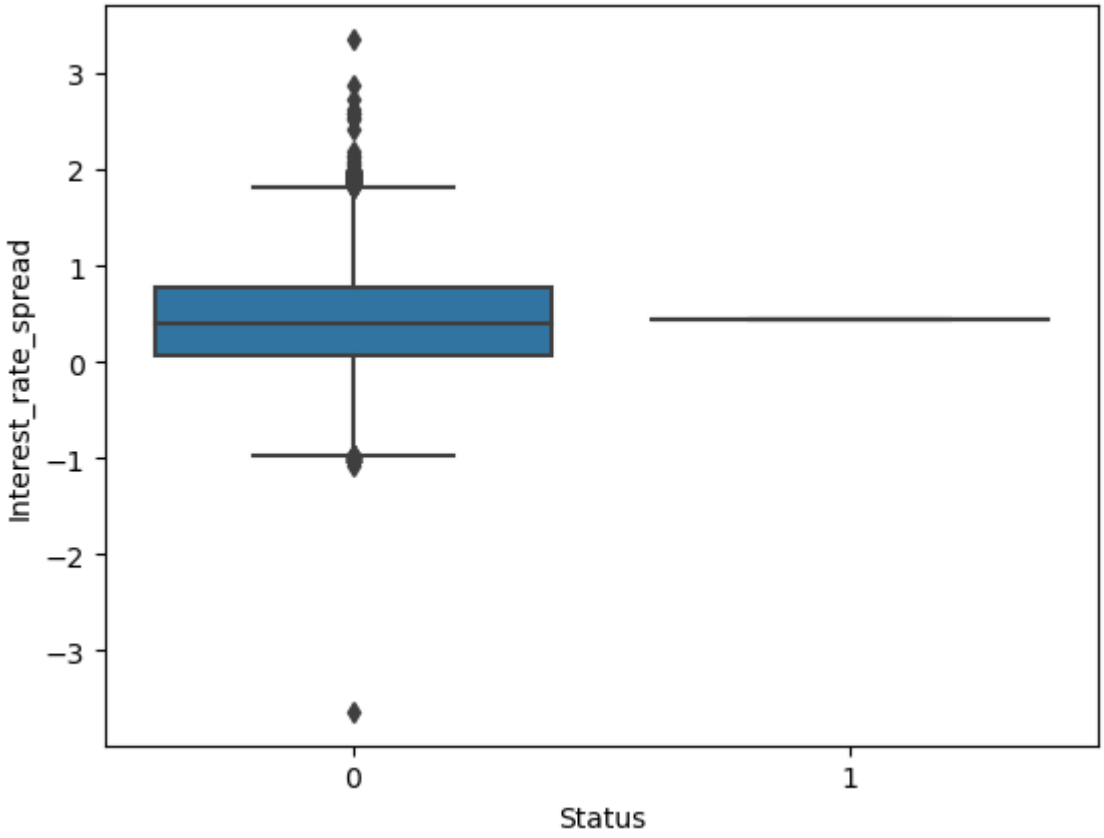
In [51]:
```
sns.histplot(df.Interest_rate_spread,binwidth=.5)
```

Out[51]:
```
<Axes: xlabel='Interest_rate_spread', ylabel='Count'>
```



In [52]:
```
sns.boxplot(x='Status',y='Interest_rate_spread',data=df)
```

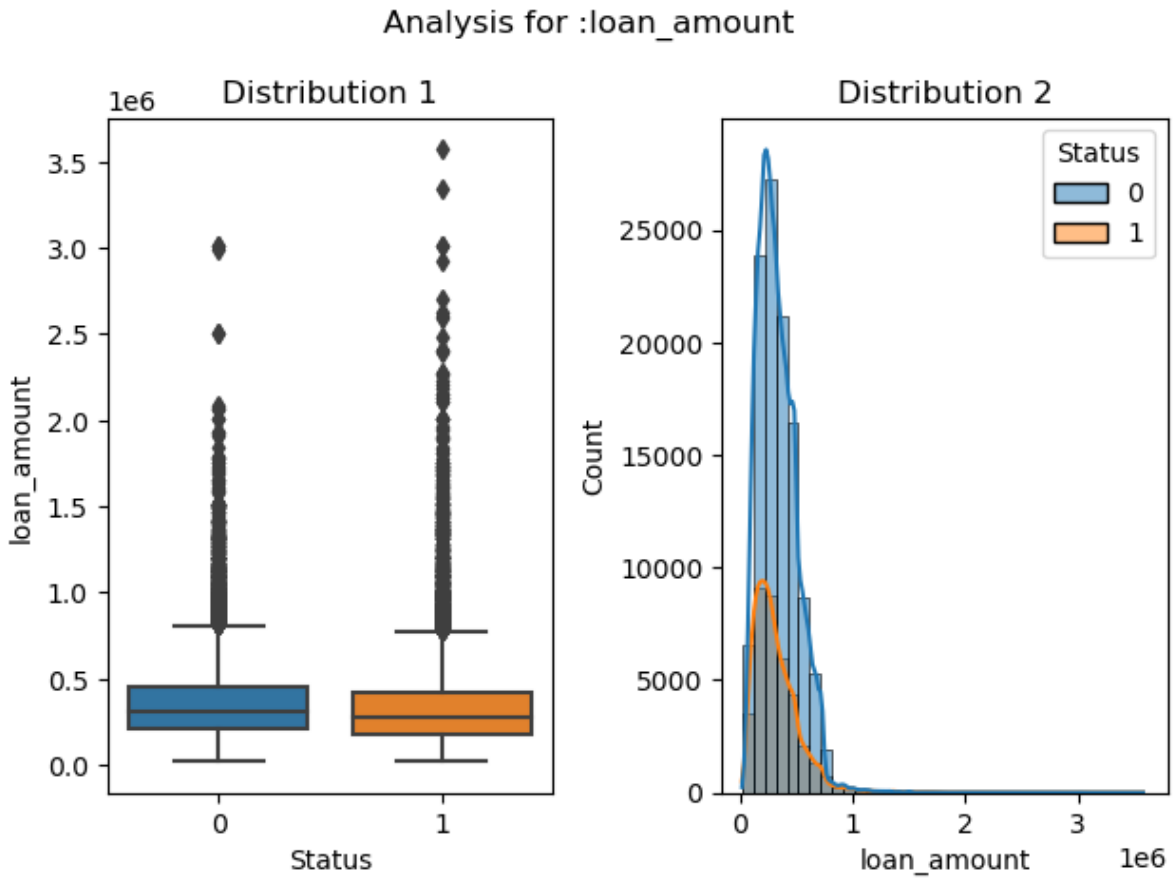
Out[52]:
```
<Axes: xlabel='Status', ylabel='Interest_rate_spread'>
```



Subplots

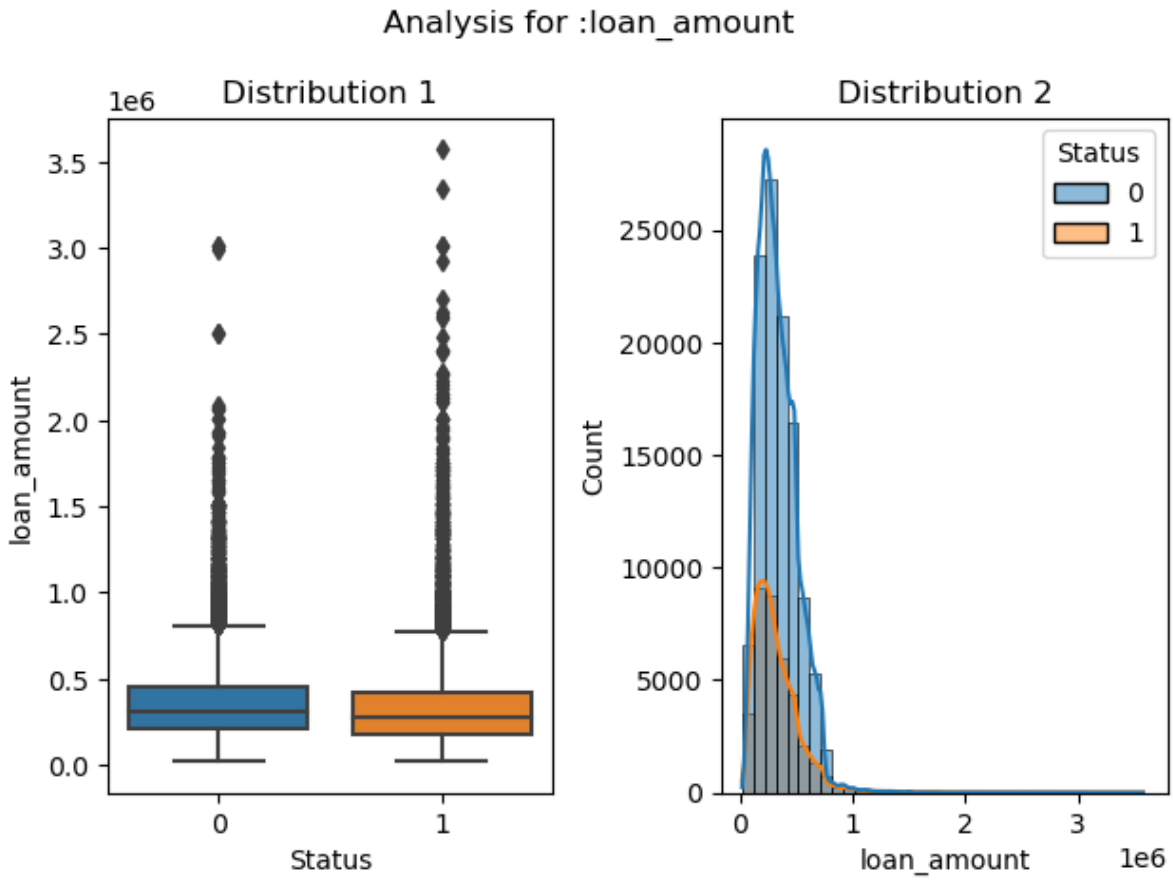
In [53]:
```
from ipywidgets import interact
```

In [55]:
```
var = 'loan_amount'
fig,ax = plt.subplots(1,2)
sns.boxplot(x='Status',y=var,data=df,ax=ax[0])
sns.histplot(x=var,data=df,ax=ax[1],kde=True,binwidth=100000,hue='Status')
ax[0].set_title('Distribution 1')
ax[1].set_title('Distribution 2')
plt.suptitle(f'Analysis for :{var}')
plt.tight_layout()
plt.show()
```

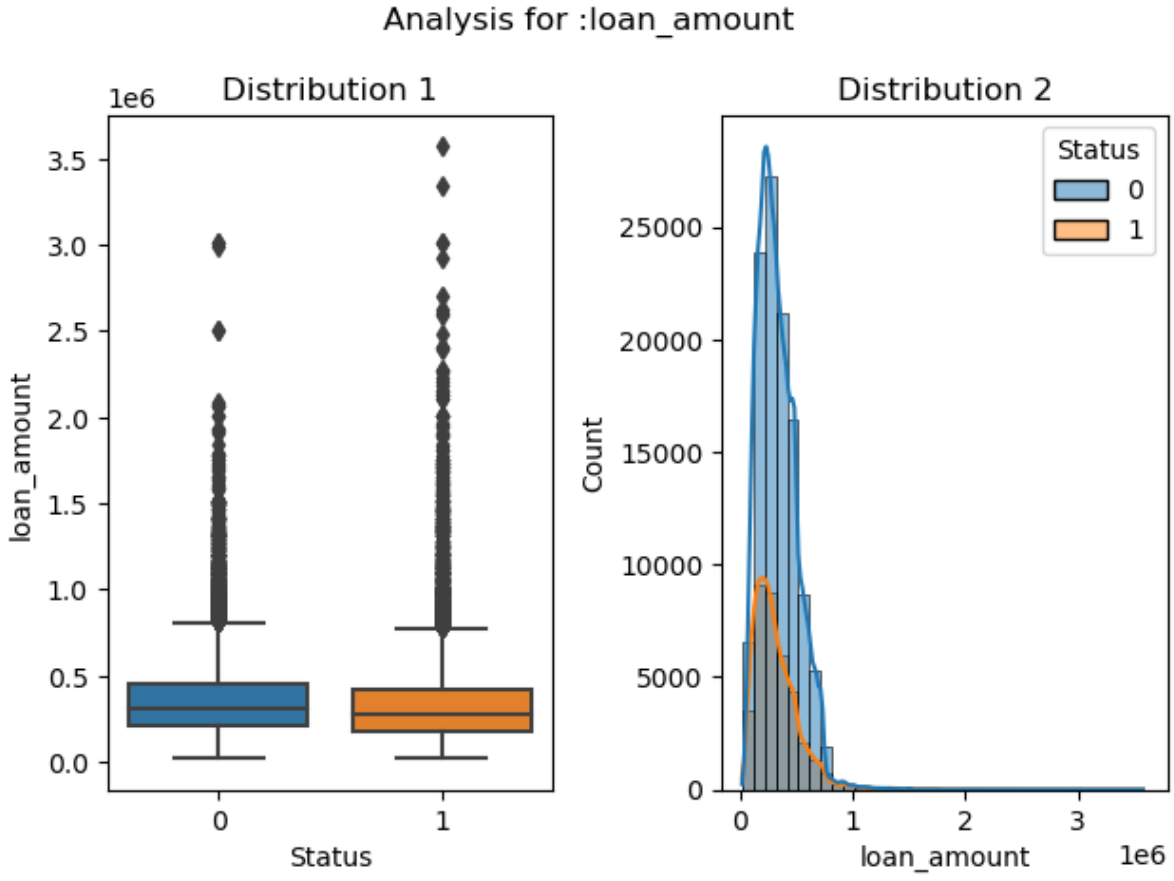


In [58]:
```
def plot_numerical_analysis(var):
    fig,ax = plt.subplots(1,2)
    sns.boxplot(x='Status',y=var,data=df,ax=ax[0])
    sns.histplot(x=var,data=df,ax=ax[1],kde=True,binwidth=100000,hue='Status')
    ax[0].set_title('Distribution 1')
    ax[1].set_title('Distribution 2')
    plt.suptitle(f'Analysis for :{var}')
    plt.tight_layout()
    plt.show()
```

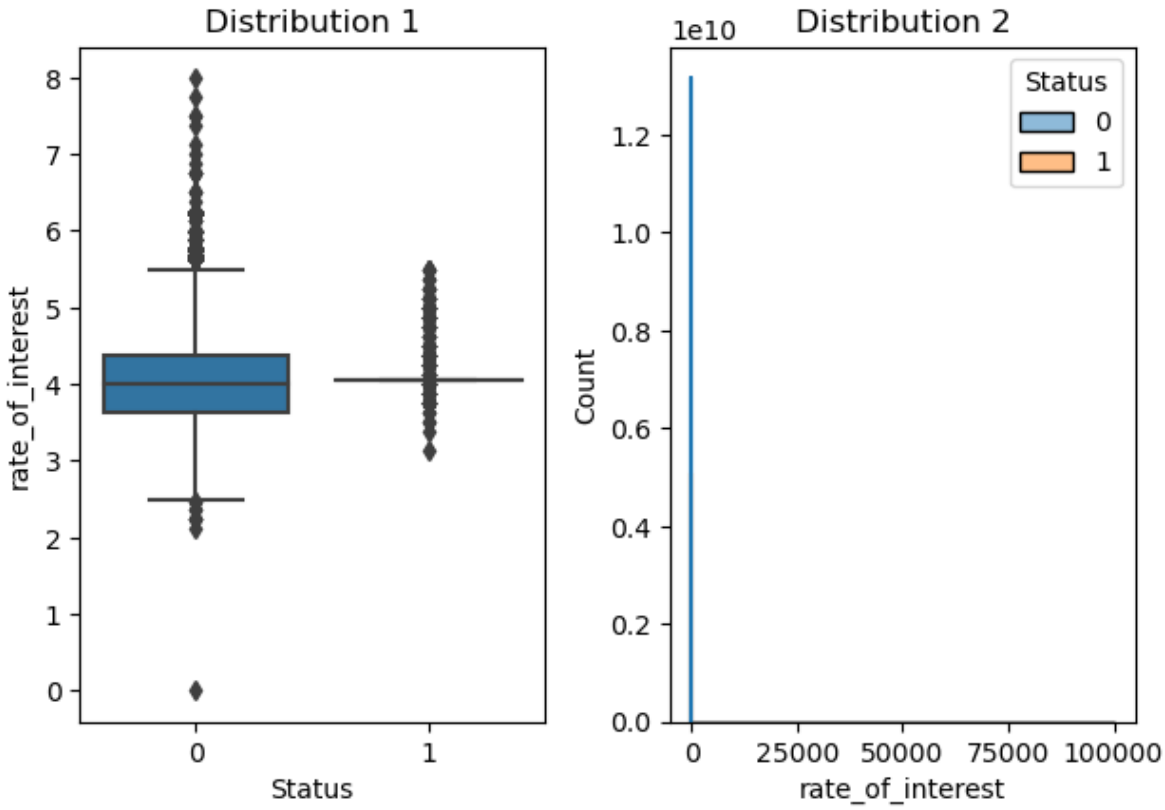
In [60]:
```
plot_numerical_analysis('loan_amount')
```



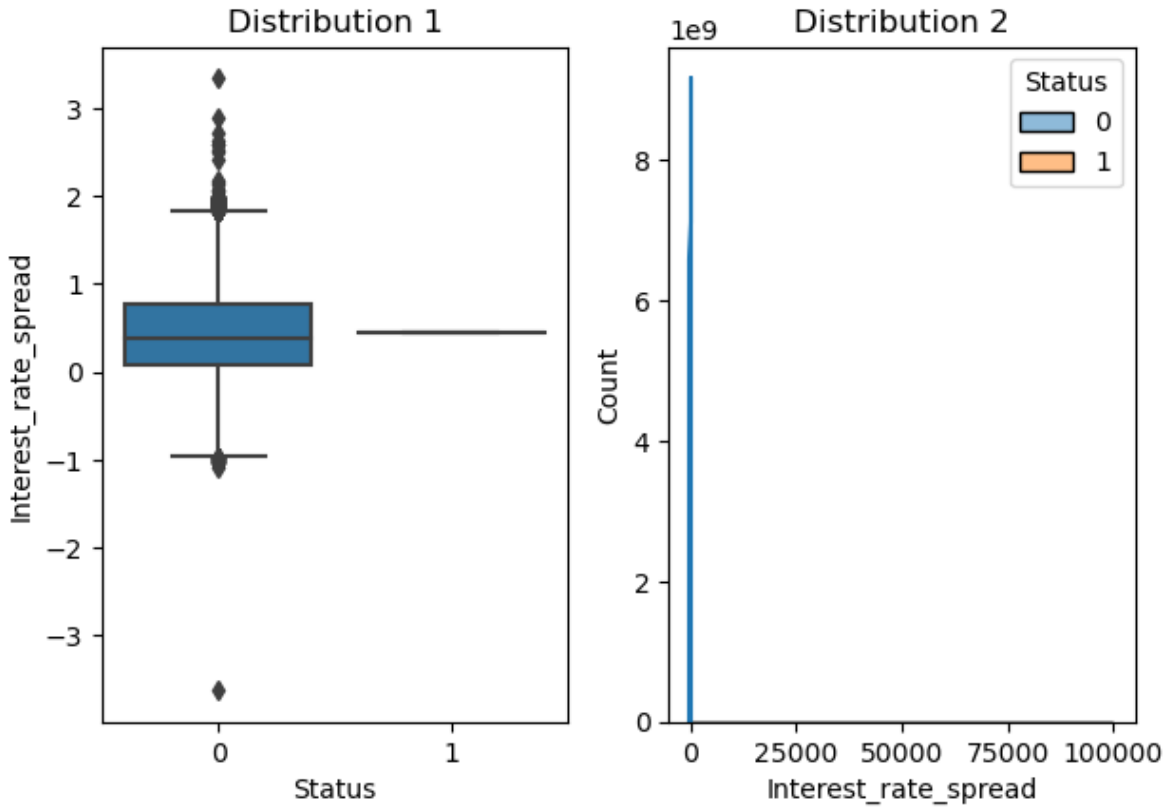
In [61]:
```
for n in num_cols[2:]:
    if n != 'Status':
        try:
            plot_numerical_analysis(n)
        except:
            print(f'{n} - Not Processed')
```

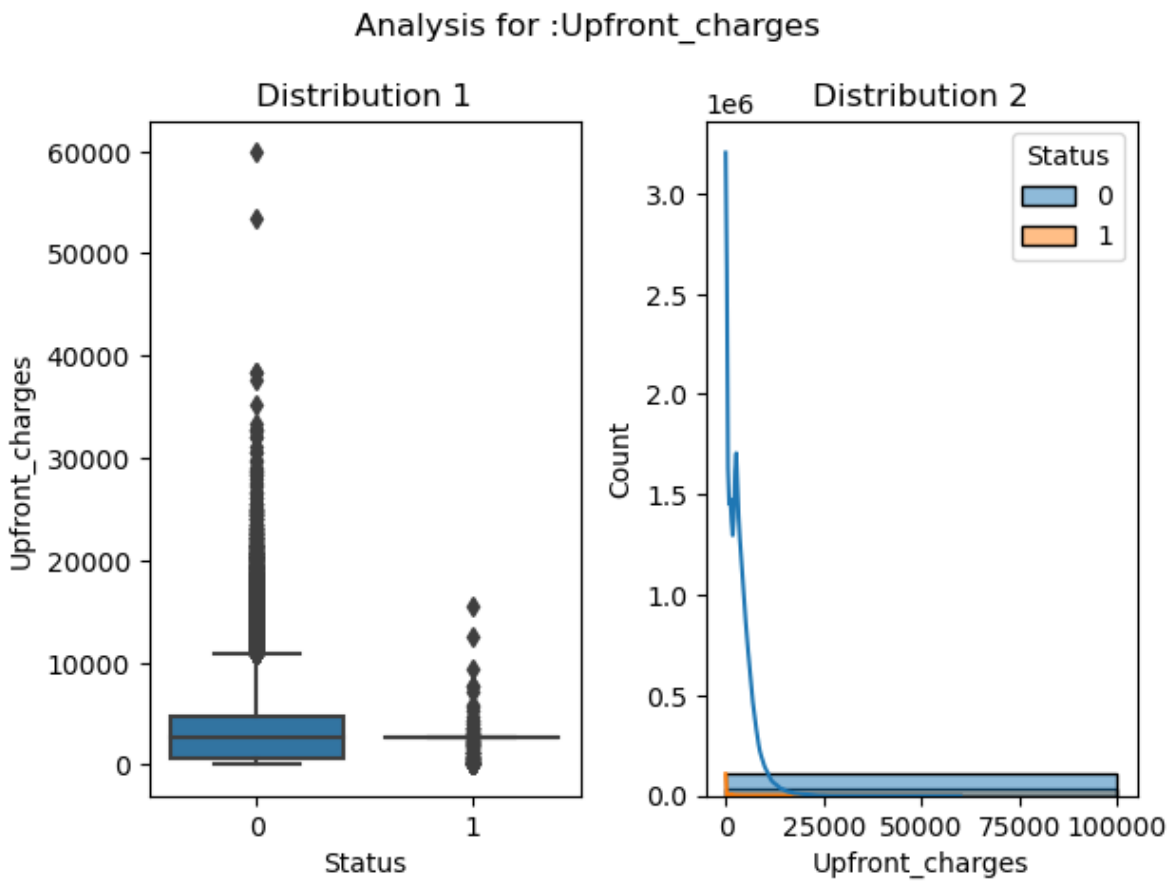


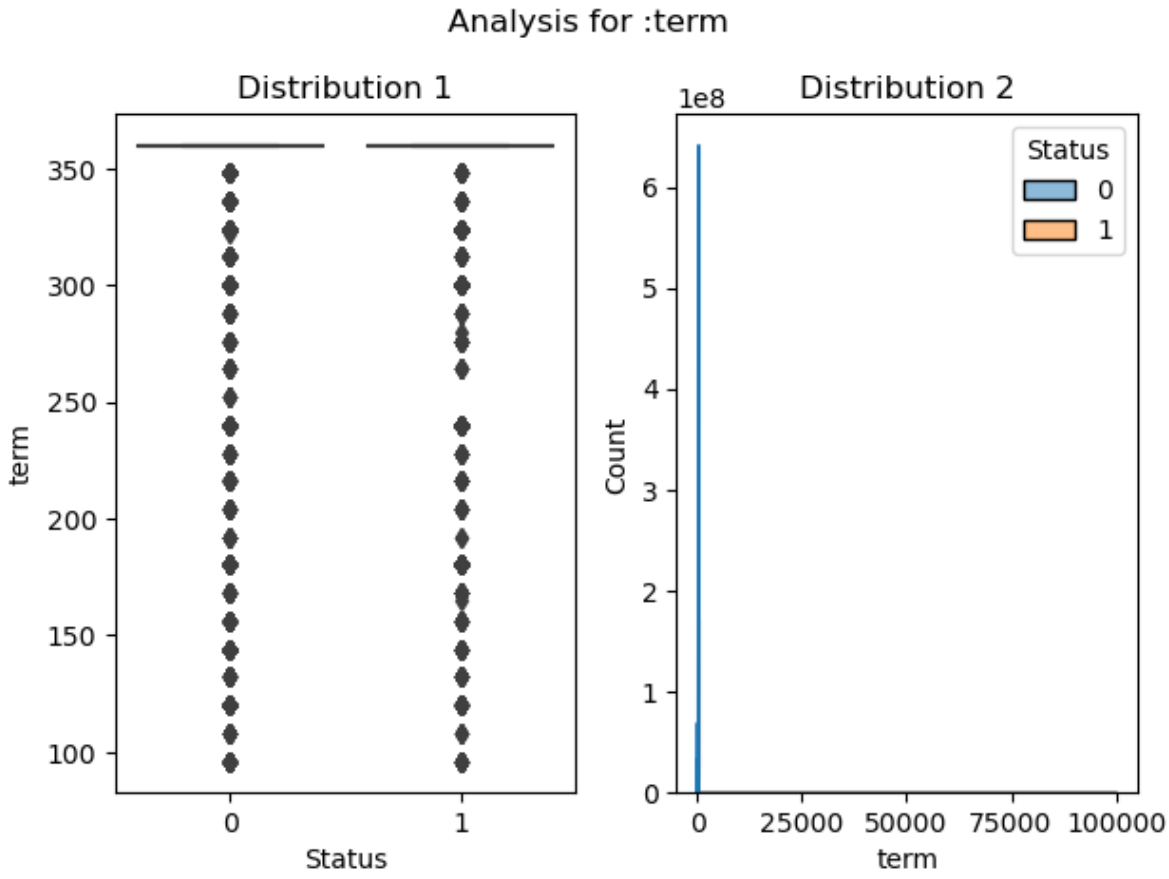
Analysis for :rate_of_interest



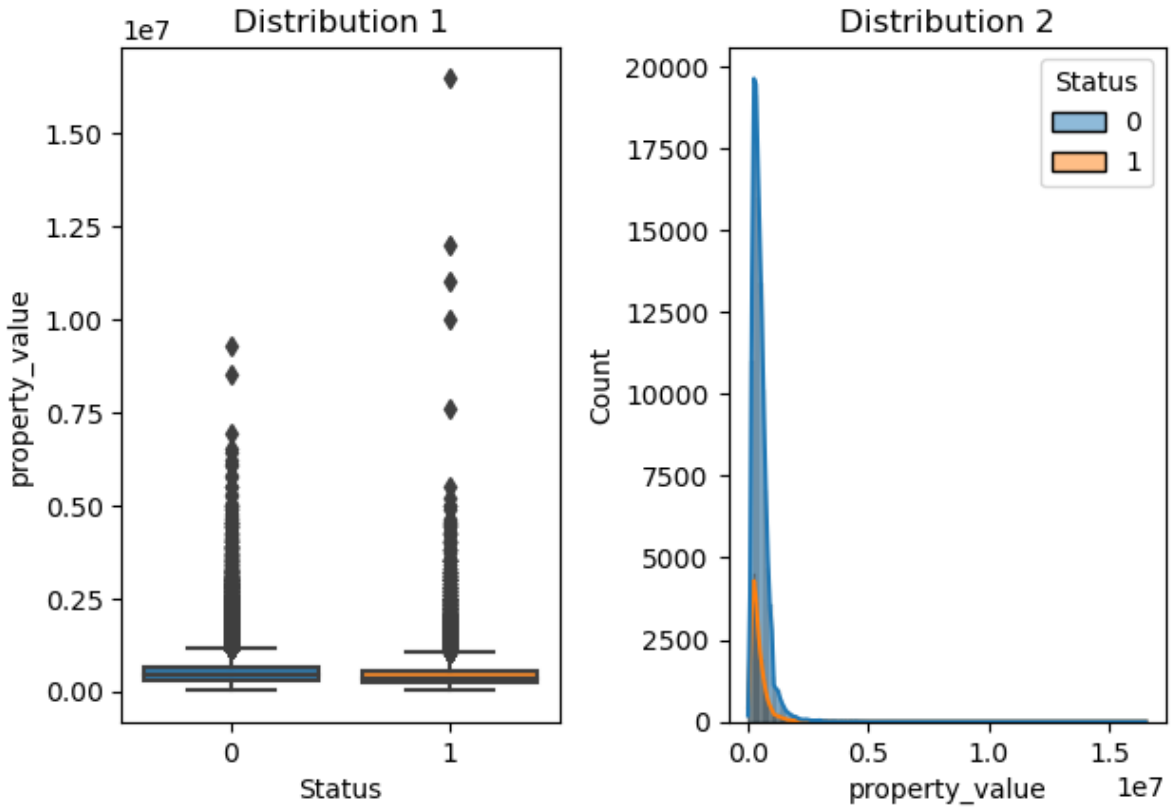
Analysis for :Interest_rate_spread



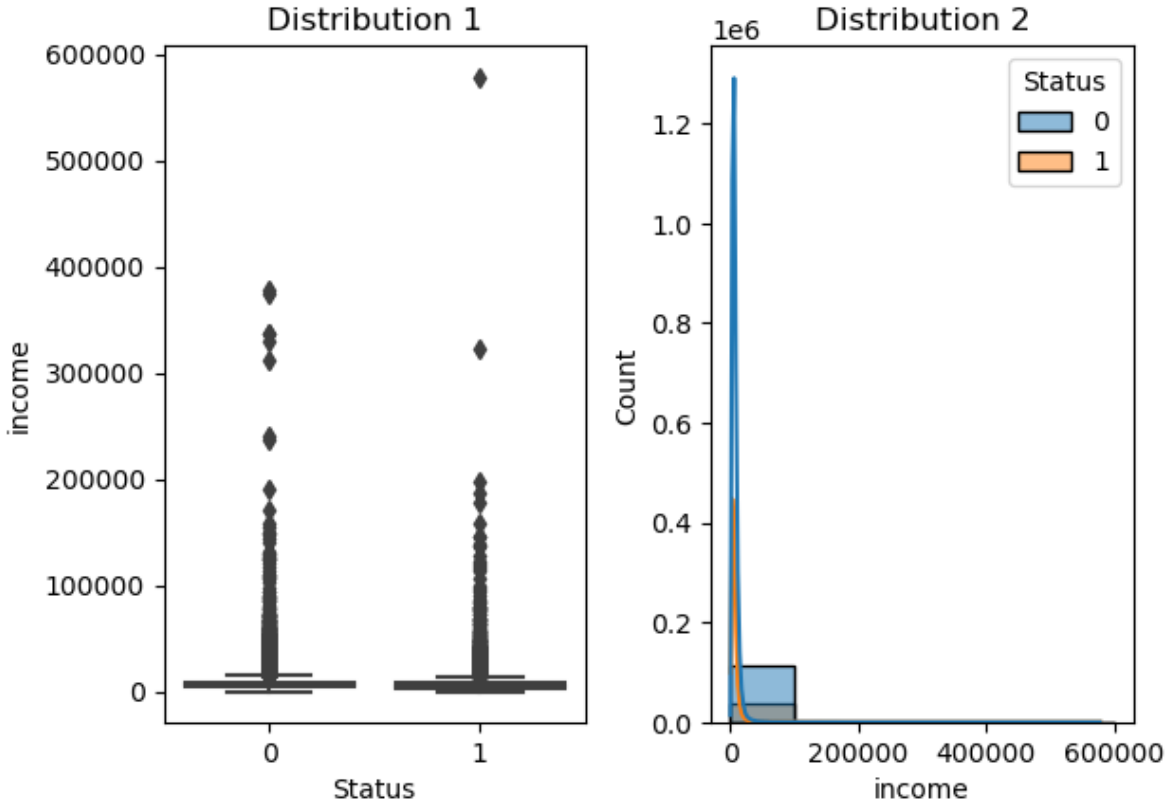

Analysis for :Upfront_charges
Distribution 1
Upfront_charges
Status
Distribution 2
1e6
Count
Upfront_charges
Status
0
1

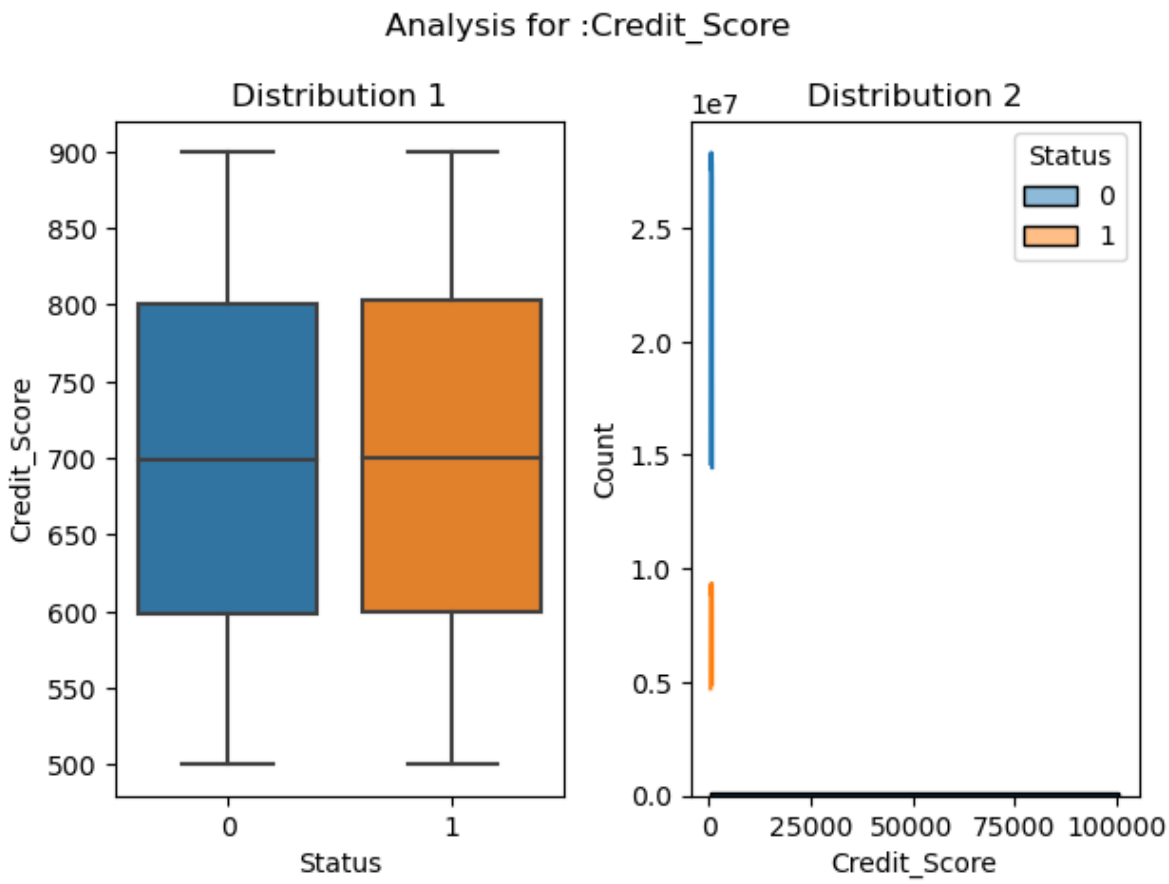

Analysis for :term
Distribution 1
term
Status
Distribution 2
1e8
Count
term
Status
0
1

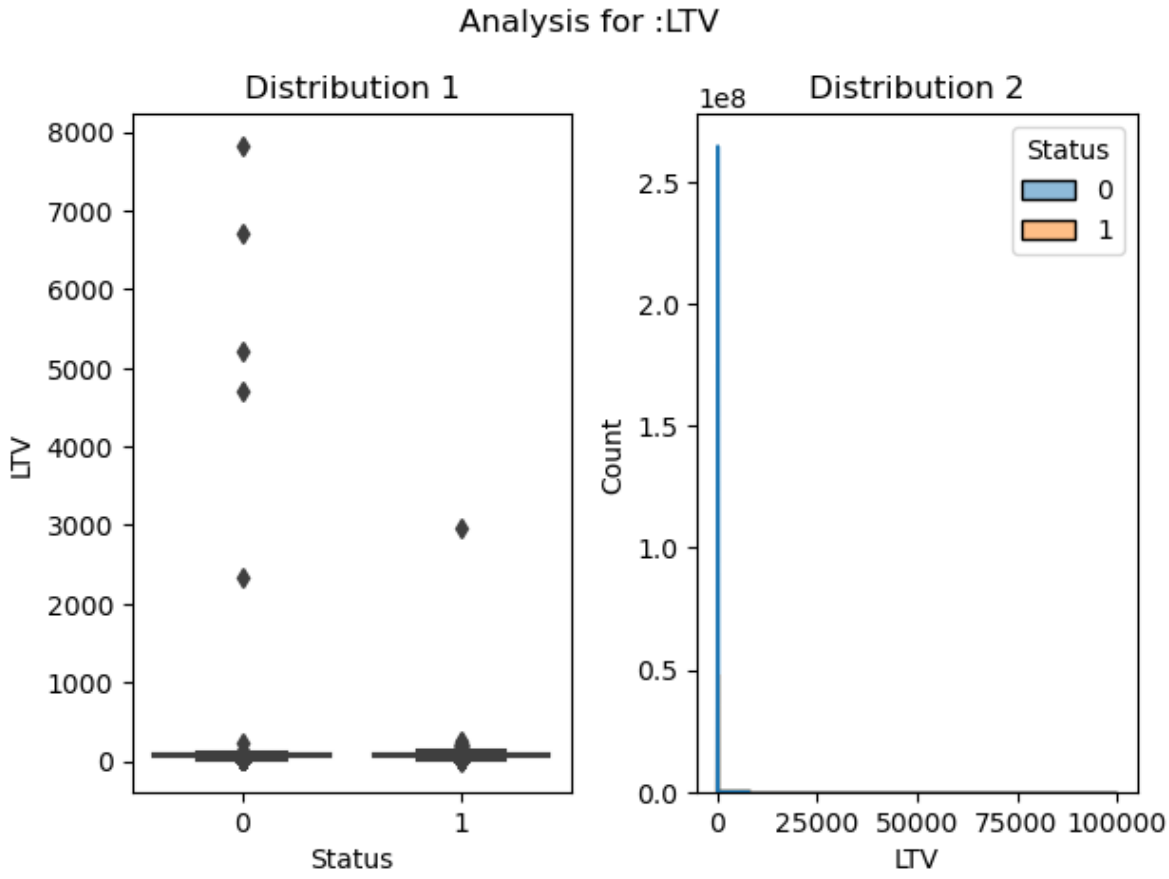
Analysis for :property_value

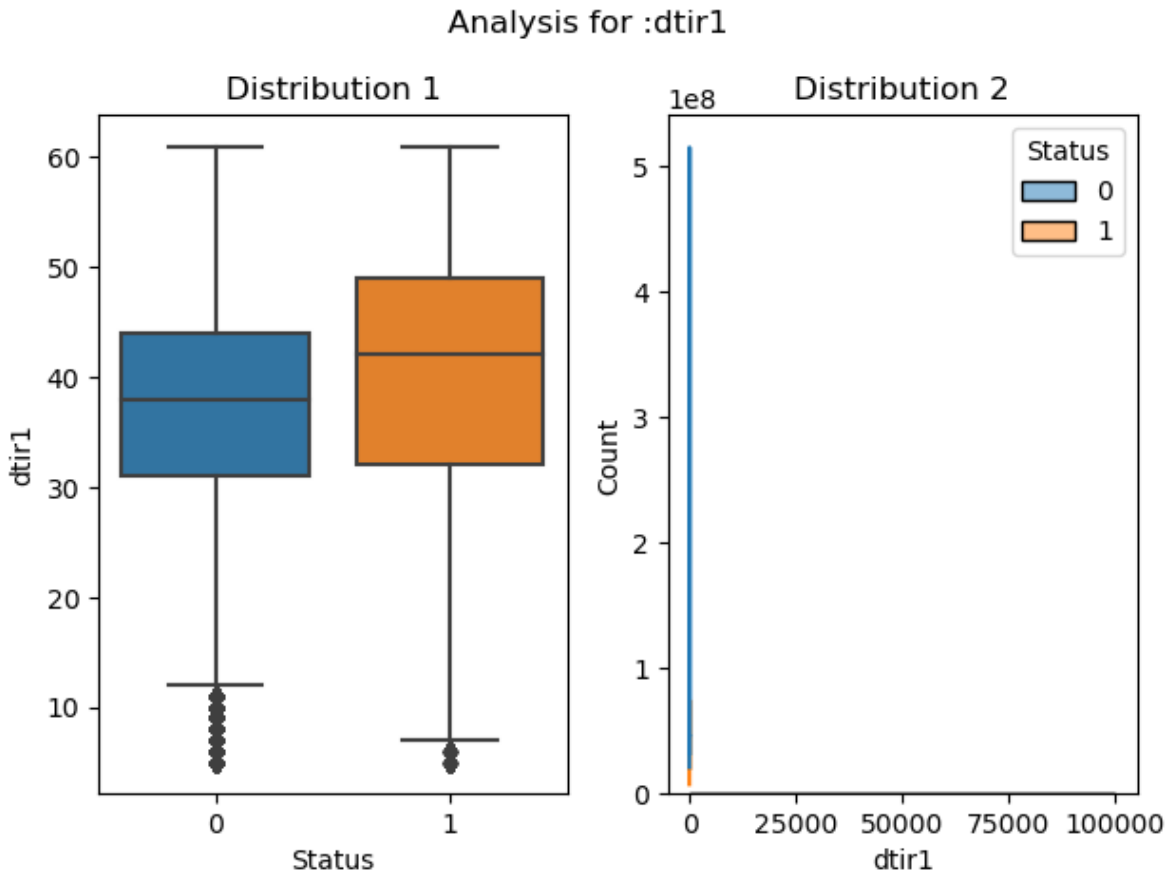


Analysis for :income




Analysis for :Credit_Score
Distribution 1
Credit_Score
900
850
800
750
700
650
600
550
500
0
1
Status
Distribution 2
1e7
Count
2.5
2.0
1.5
1.0
0.5
0.0
Status
0
1
0
25000
50000
75000
100000
Credit_Score


Analysis for :LTV
Distribution 1
LTV
8000
7000
6000
5000
4000
3000
2000
1000
0
0
1
Status
Distribution 2
1e8
Count
2.5
2.0
1.5
1.0
0.5
0.0
Status
0
1
0
25000
50000
75000
100000
LTV



In [62]:
```
interact(plot_numerical_analysis,var= num_cols)
```

Out[62]:
```
interactive(children=(Dropdown(description='var', options=('ID', 'year', 'loan_amo
unt', 'rate_of_interest', 'I…
<function __main__.plot_numerical_analysis(var)>
```

In [63]:
```
df.select_dtypes('O').columns
```

Out[63]:
```
Index(['loan_limit', 'Gender', 'approv_in_adv', 'loan_type', 'loan_purpose',
       'Credit_Worthiness', 'open_credit', 'business_or_commercial',
       'Neg_ammortization', 'interest_only', 'lump_sum_payment',
       'construction_type', 'occupancy_type', 'Secured_by', 'total_units',
       'credit_type', 'co-applicant_credit_type', 'age',
       'submission_of_application', 'Region', 'Security_Type'],
      dtype='object')
```

In [64]:
```
temp = df.groupby(['Gender','Status']).ID.count()
temp = temp.reset_index()
temp
```

Out[64]:

	Gender	Status	ID
0	Female	0	20418
1	Female	1	6848
2	Joint	0	33466
3	Joint	1	7933
4	Male	0	31255
5	Male	1	11091
6	Sex Not Available	0	26892
7	Sex Not Available	1	10767

In [65]:
```
temp = df[df['loan_limit']=='cf']
temp = temp.set_index('ID')
temp.head()
```

Out[65]:

ID	year	loan_limit	Gender	approv_in_adv	loan_type	loan_purpose	Credit_Worthiness	ope
24890	2019	cf	Sex Not Available	nopre	type1	p1	l1	
24891	2019	cf	Male	nopre	type2	p1	l1	
24892	2019	cf	Male	pre	type1	p1	l1	
24893	2019	cf	Male	nopre	type1	p4	l1	
24894	2019	cf	Joint	pre	type1	p1	l1	

5 rows × 33 columns

In [66]:
```
temp = temp.reset_index()
temp.head()
```

Out[66]:

	ID	year	loan_limit	Gender	approv_in_adv	loan_type	loan_purpose	Credit_Worthiness	
0	24890	2019	cf	Sex Not Available	nopre	type1	p1	l1	
1	24891	2019	cf	Male	nopre	type2	p1	l1	
2	24892	2019	cf	Male	pre	type1	p1	l1	
3	24893	2019	cf	Male	nopre	type1	p4	l1	
4	24894	2019	cf	Joint	pre	type1	p1	l1	

5 rows × 34 columns

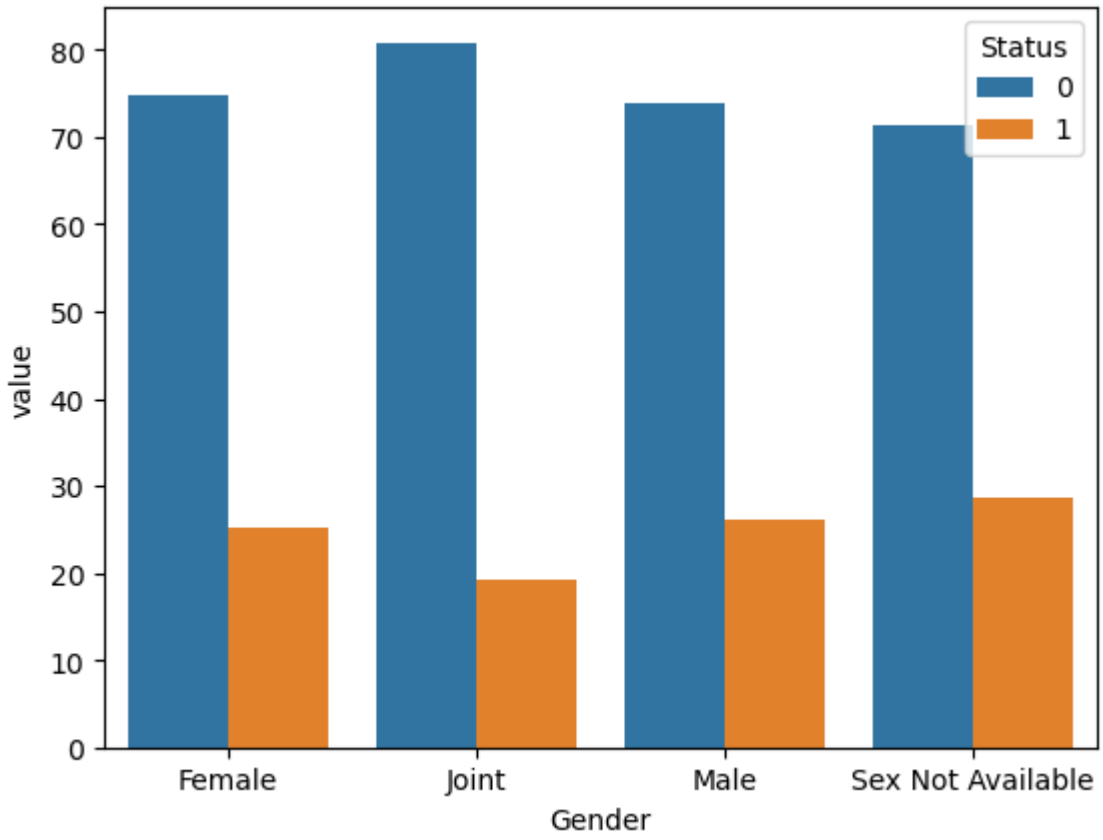
In [67]:
```
temp = pd.pivot_table(index='Gender',columns='Status',aggfunc='count',values='ID',c
temp['total']=temp.sum(axis=1)
temp[0]=temp[0]*100/temp['total']
temp[1]=temp[1]*100/temp['total']
temp.drop('total',axis=1,inplace=True)
temp = temp.round(2)
temp = temp.reset_index()
temp = temp.melt(id_vars='Gender')
temp
```

Out[67]:

	Gender	Status	value
0	Female	0	74.88
1	Joint	0	80.84
2	Male	0	73.81
3	Sex Not Available	0	71.41
4	Female	1	25.12
5	Joint	1	19.16
6	Male	1	26.19
7	Sex Not Available	1	28.59

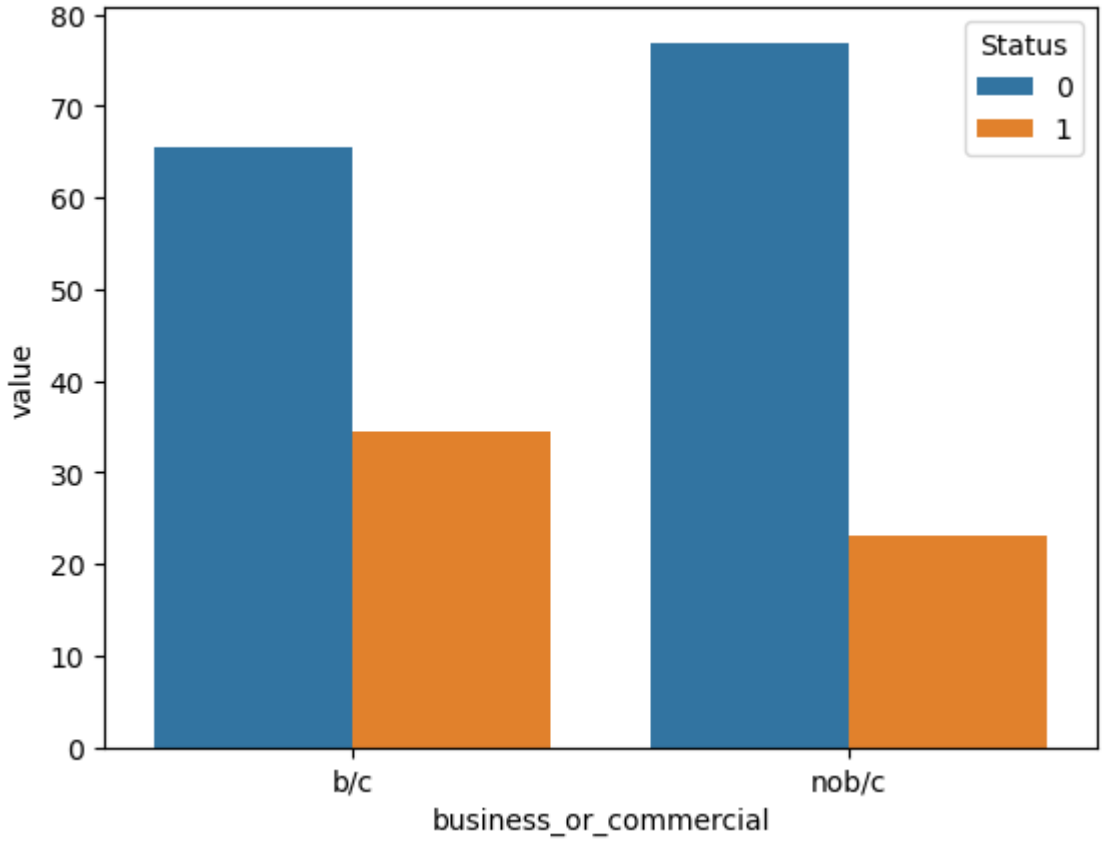
In [68]:
```
sns.barplot(x='Gender',y='value',data=temp,hue='Status')
```

Out[68]:
```
<Axes: xlabel='Gender', ylabel='value'>
```



In [69]:
```
var = 'business_or_commercial'
temp = pd.pivot_table(index=var,columns='Status',aggfunc='count',values='ID',data=d
temp['total']=temp.sum(axis=1)
temp[0]=temp[0]*100/temp['total']
temp[1]=temp[1]*100/temp['total']
temp.drop('total',axis=1,inplace=True)
temp = temp.round(2)
temp = temp.reset_index()
temp = temp.melt(id_vars=var)
sns.barplot(x=var,y='value',data=temp,hue='Status')
```

Out[69]:
```
<Axes: xlabel='business_or_commercial', ylabel='value'>
```



In [70]:
```
def plot_car_analysis(var):
    temp = pd.pivot_table(index=var,columns='Status',aggfunc='count',values='ID',da
    temp['total']=temp.sum(axis=1)
    temp[0]=temp[0]*100/temp['total']
    temp[1]=temp[1]*100/temp['total']
    temp.drop('total',axis=1,inplace=True)
    temp = temp.round(2)
    temp = temp.reset_index()
    temp = temp.melt(id_vars=var)
    sns.barplot(x=var,y='value',data=temp,hue='Status')
```

In [71]:
```
interact(plot_car_analysis,var=df.select_dtypes('O').columns)
```

Out[71]:
```
interactive(children=(Dropdown(description='var', options=('loan_limit', 'Gender',
'approv_in_adv', 'loan_type…
<function __main__.plot_car_analysis(var)>
```

In [83]:
```
temp = pd.pivot_table(index='loan_type',columns='Status',values='ID',aggfunc='count
temp = temp.reset_index()
temp.melt(id_vars='loan_type')
```

Out[83]:

	loan_type	Status	value
0	type1	0	87398
1	type2	0	13590
2	type3	0	11043
3	type1	1	25775
4	type2	1	7172
5	type3	1	3692

In [84]:
```
temp = pd.pivot_table(index=['loan_type','Gender'],columns='Status',values='ID',agg
temp = temp.reset_index()
temp.melt(id_vars=['loan_type','Gender'])
```

Out[84]:

	loan_type	Gender	Status	value
0	type1	Female	0	16866
1	type1	Joint	0	26023
2	type1	Male	0	23980
3	type1	Sex Not Available	0	20529
4	type2	Female	0	2947
5	type2	Joint	0	3654
6	type2	Male	0	3642
7	type2	Sex Not Available	0	3347
8	type3	Female	0	605
9	type3	Joint	0	3789
10	type3	Male	0	3633
11	type3	Sex Not Available	0	3016
12	type1	Female	1	4981
13	type1	Joint	1	5739
14	type1	Male	1	7611
15	type1	Sex Not Available	1	7444
16	type2	Female	1	1597
17	type2	Joint	1	1272
18	type2	Male	1	2059
19	type2	Sex Not Available	1	2244
20	type3	Female	1	270
21	type3	Joint	1	922
22	type3	Male	1	1421
23	type3	Sex Not Available	1	1079

In [ ]:

In [ ]: