In [2]:
```
import pandas as pd
import numpy as np
import re
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import datetime as dt
import dateutil.relativedelta as rd
from sklearn.model_selection import train_test_split as split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
plt.style.use('ggplot')
```

In [3]:
```
data = pd.read_excel("C:\\Users\\mahi2\\Desktop\\Project2_Dataset (1)\\Dataset\data.xl
```

In [4]:
```
data
```

Out[4]:

	UniqueID	disbursed_amount	asset_cost	ltv	branch_id	supplier_id	manufacturer_id	Curr
0	420825	50578	58400	89.55	67	22807	45	
1	417566	53278	61360	89.63	67	22807	45	
2	539055	52378	60300	88.39	67	22807	45	
3	529269	46349	61500	76.42	67	22807	45	
4	563215	43594	78256	57.50	67	22744	86	
...	...	...	...	...	...	...	...	
233149	561031	57759	76350	77.28	5	22289	51	
233150	649600	55009	71200	78.72	138	17408	51	
233151	603445	58513	68000	88.24	135	23313	45	
233152	442948	22824	40458	61.79	160	16212	48	
233153	545300	35299	72698	52.27	3	14573	45	

233154 rows × 41 columns

preliminary analysis

Perform preliminary data inspection and report the findings as the structure of the data, missing values, duplicates etc. The variable names in the data are not in accordance with the identifier naming in Python. Change the variable names accordingly.

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 233154 entries, 0 to 233153
Data columns (total 41 columns):
 #   Column                               Non-Null Count   Dtype
---  ------                               --------------   -----
 0   UniqueID                             233154 non-null  int64
 1   disbursed_amount                     233154 non-null  int64
 2   asset_cost                           233154 non-null  int64
 3   ltv                                  233154 non-null  float64
 4   branch_id                            233154 non-null  int64
 5   supplier_id                          233154 non-null  int64
 6   manufacturer_id                      233154 non-null  int64
 7   Current_pincode_ID                   233154 non-null  int64
 8   Date.of.Birth                        233154 non-null  datetime64[ns]
 9   Employment.Type                      225493 non-null  object
 10  DisbursalDate                        233154 non-null  datetime64[ns]
 11  State_ID                             233154 non-null  int64
 12  Employee_code_ID                     233154 non-null  int64
 13  MobileNo_Avl_Flag                    233154 non-null  int64
 14  Aadhar_flag                          233154 non-null  int64
 15  PAN_flag                             233154 non-null  int64
 16  VoterID_flag                         233154 non-null  int64
 17  Driving_flag                         233154 non-null  int64
 18  Passport_flag                        233154 non-null  int64
 19  PERFORM_CNS.SCORE                    233154 non-null  int64
 20  PERFORM_CNS.SCORE.DESCRIPTION        233154 non-null  object
 21  PRI.NO.OF.ACCTS                      233154 non-null  int64
 22  PRI.ACTIVE.ACCTS                     233154 non-null  int64
 23  PRI.OVERDUE.ACCTS                    233154 non-null  int64
 24  PRI.CURRENT.BALANCE                  233154 non-null  int64
 25  PRI.SANCTIONED.AMOUNT                233154 non-null  int64
 26  PRI.DISBURSED.AMOUNT                 233154 non-null  int64
 27  SEC.NO.OF.ACCTS                      233154 non-null  int64
 28  SEC.ACTIVE.ACCTS                     233154 non-null  int64
 29  SEC.OVERDUE.ACCTS                    233154 non-null  int64
 30  SEC.CURRENT.BALANCE                  233154 non-null  int64
 31  SEC.SANCTIONED.AMOUNT                233154 non-null  int64
 32  SEC.DISBURSED.AMOUNT                 233154 non-null  int64
 33  PRIMARY.INSTAL.AMT                   233154 non-null  int64
 34  SEC.INSTAL.AMT                       233154 non-null  int64
 35  NEW.ACCTS.IN.LAST.SIX.MONTHS         233154 non-null  int64
 36  DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS  233154 non-null  int64
 37  AVERAGE.ACCT.AGE                     233154 non-null  object
 38  CREDIT.HISTORY.LENGTH                233154 non-null  object
 39  NO.OF_INQUIRIES                      233154 non-null  int64
 40  loan_default                         233154 non-null  int64
dtypes: datetime64[ns](2), float64(1), int64(34), object(4)
memory usage: 72.9+ MB
```

column names need to be changed

In [6]:
```
data.columns = data.columns.str.replace('.','_').str.lower()
```

In [7]:
```
print('Are there any duplicated rows ??',data.duplicated().any())
```

```
Are there any duplicated rows ?? False
```

missing values

In [8]:
```
print('Missing values in variable')
print(data.isnull().sum())
```

```
Missing values in variable
uniqueid                                  0
disbursed_amount                          0
asset_cost                                0
ltv                                       0
branch_id                                 0
supplier_id                               0
manufacturer_id                           0
current_pincode_id                        0
date_of_birth                             0
employment_type                        7661
disbursaldate                             0
state_id                                  0
employee_code_id                          0
mobileno_avl_flag                         0
aadhar_flag                               0
pan_flag                                  0
voterid_flag                              0
driving_flag                              0
passport_flag                             0
perform_cns_score                         0
perform_cns_score_description             0
pri_no_of_accts                           0
pri_active_accts                          0
pri_overdue_accts                         0
pri_current_balance                       0
pri_sanctioned_amount                     0
pri_disbursed_amount                      0
sec_no_of_accts                           0
sec_active_accts                          0
sec_overdue_accts                         0
sec_current_balance                       0
sec_sanctioned_amount                     0
sec_disbursed_amount                      0
primary_instal_amt                        0
sec_instal_amt                            0
new_accts_in_last_six_months              0
delinquent_accts_in_last_six_months       0
average_acct_age                          0
credit_history_length                     0
no_of_inquiries                           0
loan_default                              0
dtype: int64
```

In [9]:
```
#Checking Missing Values
missing_vars = pd.DataFrame(data.isnull().sum())
missing_vars.columns = ["count"]
missing_vars.loc[missing_vars["count"] > 0]
```

Out[9]:

	count
employment_type	7661

Provide the statistical description of the numerical data variables¶

exploring the unique ids

In [10]:
```
id_data =[  'uniqueid', 'branch_id', 'state_id', 'manufacturer_id', 'current_pincode_i
```

In [11]:
```
for i in id_data:
    print('No. of Unique values for {} : \n{}'.format(i.upper(),data[i].nunique()) )
```

```
No. of Unique values for UNIQUEID :
233154
No. of Unique values for BRANCH_ID :
82
No. of Unique values for STATE_ID :
22
No. of Unique values for MANUFACTURER_ID :
11
No. of Unique values for CURRENT_PINCODE_ID :
6698
No. of Unique values for STATE_ID :
22
No. of Unique values for EMPLOYEE_CODE_ID :
3270
```

Overall Statistical Description

identifying categorical data :

In [12]:
```
cat_cols = ['branch_id', 'supplier_id', 'manufacturer_id', 'state_id',
            'new_accts_in_last_six_months','delinquent_accts_in_last_six_months',
            'pri_no_of_accts', 'pri_active_accts','pri_overdue_accts',
            'sec_no_of_accts', 'sec_active_accts','sec_overdue_accts','average_acct_ag
for i in cat_cols :
    data[i] = data[i].astype('category')
```

In [13]:
```
binary_columns = list(data.nunique()[data.nunique() == 2].index)
```

description for categorical columns

In [14]:
```
data.describe(include = 'category')
```

Out[14]:

	branch_id	supplier_id	manufacturer_id	state_id	pri_no_of_accts	pri_active_accts	pri_overdue
count	233154	233154	233154	233154	233154	233154	2
unique	82	2953	11	22	108	40	
top	2	18317	86	4	0	0	
freq	13138	1432	109534	44870	116950	137016	2

description for binary data using value counts

In [15]:
```
data.describe(include = 'category')
```

Out[15]:

	branch_id	supplier_id	manufacturer_id	state_id	pri_no_of_accts	pri_active_accts	pri_overdue
count	233154	233154	233154	233154	233154	233154	2
unique	82	2953	11	22	108	40	
top	2	18317	86	4	0	0	
freq	13138	1432	109534	44870	116950	137016	2

description for binary data using value counts

In [16]:
```
for i in binary_columns:
    vc = data[i].value_counts()
    print(i.replace('_',' ').upper(), ':')
    for j in vc.index :
        print(j ,':', vc[j])
```

```
EMPLOYMENT TYPE :
Self employed : 127635
Salaried : 97858
AADHAR FLAG :
1 : 195924
0 : 37230
PAN FLAG :
0 : 215533
1 : 17621
VOTERID FLAG :
0 : 199360
1 : 33794
DRIVING FLAG :
0 : 227735
1 : 5419
PASSPORT FLAG :
0 : 232658
1 : 496
LOAN DEFAULT :
0 : 182543
1 : 50611
```

describe quantitative data

In [17]:
```
col = cat_cols + binary_columns

quant = data.loc[:,~data.columns.isin(col)]

quant.describe().loc[['min','25%','mean','50%','75%','max','std']].round(1)
```

Out[17]:

	uniqueid	disbursed_amount	asset_cost	ltv	current_pincode_id	employee_code_id	mobileno_
min	417428.0	13320.0	37000.0	10.0	1.0	1.0	
25%	476786.2	47145.0	65717.0	68.9	1511.0	713.0	
mean	535917.6	54357.0	75865.1	74.7	3396.9	1549.5	
50%	535978.5	53803.0	70946.0	76.8	2970.0	1451.0	
75%	595039.8	60413.0	79201.8	83.7	5677.0	2362.0	
max	671084.0	990572.0	1628992.0	95.0	7345.0	3795.0	
std	68315.7	12971.3	18944.8	11.5	2238.1	975.3	

unique Values in the data

In [18]:
```
data.nunique()
```

Out[18]:

```
uniqueid                               233154
disbursed_amount                        24565
asset_cost                              46252
ltv                                      6579
branch_id                                  82
supplier_id                              2953
manufacturer_id                            11
current_pincode_id                       6698
date_of_birth                           15433
employment_type                             2
disbursaldate                              84
state_id                                   22
employee_code_id                         3270
mobileno_avl_flag                           1
aadhar_flag                                 2
pan_flag                                    2
voterid_flag                                2
driving_flag                                2
passport_flag                               2
perform_cns_score                         573
perform_cns_score_description              20
pri_no_of_accts                           108
pri_active_accts                           40
pri_overdue_accts                          22
pri_current_balance                     71341
pri_sanctioned_amount                   44390
pri_disbursed_amount                    47909
sec_no_of_accts                            37
sec_active_accts                           23
sec_overdue_accts                           9
sec_current_balance                      3246
sec_sanctioned_amount                    2223
sec_disbursed_amount                     2553
primary_instal_amt                      28067
sec_instal_amt                           1918
new_accts_in_last_six_months               26
delinquent_accts_in_last_six_months        14
average_acct_age                          192
credit_history_length                     294
no_of_inquiries                            25
loan_default                                2
dtype: int64
```

How is the target variable distributed overall?

In [19]:

```
data.loan_default.value_counts()
```

Out[19]:

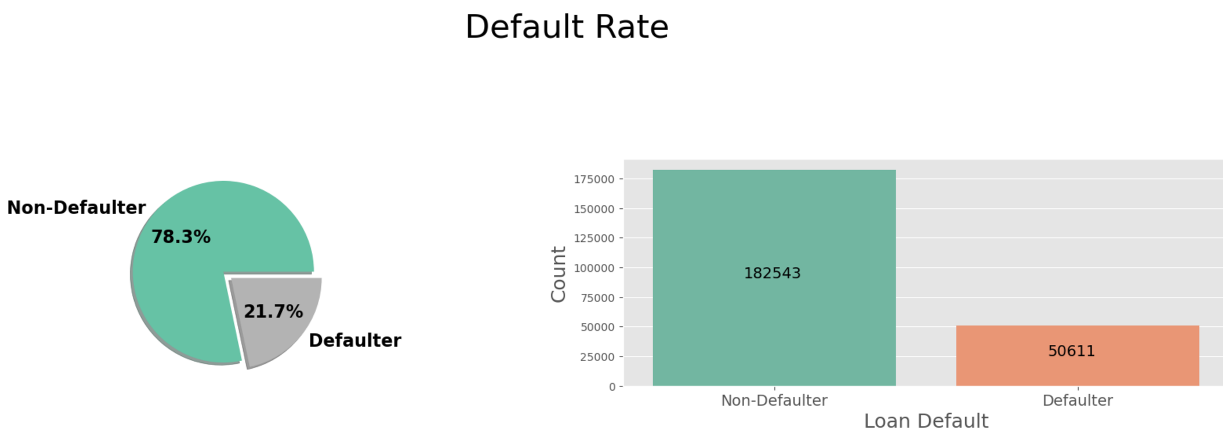
```
0    182543
1     50611
Name: loan_default, dtype: int64
```

In [20]:

```
def transform(x):
    if x == 1: return 'Defaulter'
    if x == 0: return 'Non-Defaulter'
data['loan_default_text'] = data.loan_default.apply(transform)
```

In [21]:
```
f, axes = plt.subplots(1,2, figsize = (18,6))
vc = data.loan_default_text.value_counts()
vc.plot.pie(ax = axes[0], radius = 1, cmap = 'Set2' , explode = [0.01,0.1], shadow = T
           textprops = {'color': 'black','weight': 'bold','size': 16}, )
axes[0].set_ylabel('')

sns.countplot(x='loan_default_text', data = data, ax = axes[1], palette='Set2')
for i in range(len(vc)):
    axes[1].annotate(str(vc[i]), (i-0.1,(vc[i]/2)), fontsize = 14)
axes[1].set_ylim(0,axes[1].set_ylim()[1]+5)
axes[1].set_xlabel('Loan Default',fontsize = 18)
axes[1].set_ylabel('Count',fontsize = 18)
axes[1].set_xticklabels(axes[1].get_xticklabels(),fontsize = 14)
f.suptitle('Default Rate\n', fontsize = 30)
plt.tight_layout(pad = 4)
plt.show()
```



Study the distribution of the target variable across the various categories such as branch, city, state, branch, supplier, manufacturer etc.

Univariate Analysis and Variable vs Target

In [22]:
```
def barplot(var):
    var_name = var.replace('_',' ').title()
    plt.figure(figsize = (20,5))
    sns.countplot(x='loan_default_text',data = data, palette='Set2')
    plt.title(var_name+'\n',family='Times New Roman', weight ='bold',fontsize= 25)
    plt.tight_layout()
    plt.xlabel(var_name,family='Times New Roman',fontsize= 16)
    plt.ylabel('Frequency',family='georgia',fontsize= 16)
    plt.show()
```

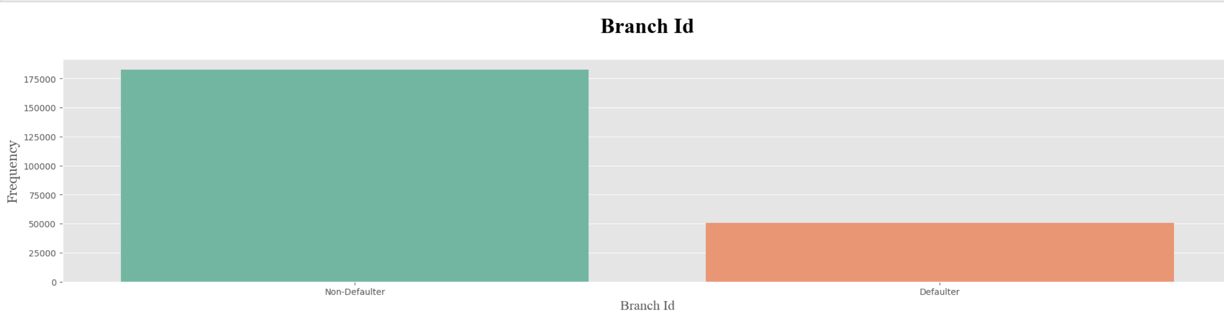
In [23]:
```
def cat_vs_target(var):
    var_name = var.replace('_',' ').title()
    plt.figure(figsize = (20,5))
    sns.countplot(x='loan_default_text',hue = 'loan_default_text', data = data, palett
```

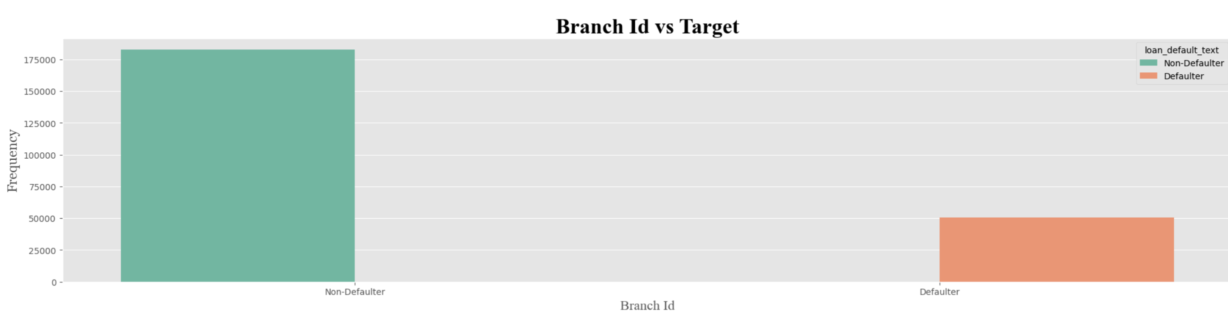
```
    plt.title(var_name + ' vs Target',family='Times New Roman', weight ='bold',fontsiz
    plt.tight_layout()
    plt.xlabel(var_name ,family='Times New Roman',fontsize= 16)
    plt.ylabel('Frequency',family='georgia',fontsize= 16)
    plt.show()
```

In [24]:
```
# is the category and target dependent on each other??
def chi_test(var):
    ct = pd.crosstab(data[var], data.loan_default_text)
    st, p, df, ef = stats.chi2_contingency(ct)
    var_name = var.replace('_',' ')
    if p >= 0.05:
        text = ('{} and Target are independent'.format(var_name.title()))
    else :
        text = ('{} and Target are dependent'.format(var_name.title()))
    plt.figure(figsize = (1,1))
    plt.plot([0,0],[0,0])
    plt.xlim(0,50)
    plt.ylim(0,5)
    plt.axis('off')
    plt.annotate(text,xy = (2.5,2.5), fontsize= 25 )
    plt.show()
```

1. branch_id

In [25]:
```
var = 'branch_id'
barplot(var)
cat_vs_target(var)
chi_test(var)
```





Branch Id and Target are dependent

Supplier Ids

In [26]:
```
var = 'supplier_id'
barplot(var)
cat_vs_target(var)
chi_test(var)
```

Supplier Id and Target are dependent

What are the different employment type given in the data? Can a strategy be developed to fill in the missing values (if any)? How does employment type define defaulters and non-defaulters? Use pie charts to express.

For Missing values : Employment Type

Only employment_type var has nulls

In [27]:
```
# Checking unique values
print("Distinct Emp Type :",data.employment_type.unique())
```

```
#Checking missing valus in percentage
print("Missing Emp Type {:.2f} %".format(data.employment_type.isnull().sum() / len(dat
```

```
Distinct Emp Type : ['Salaried' 'Self employed' nan]
Missing Emp Type 3.29 %
```

Few ways to fill in the missing values :¶

- Remove the rows as it is a very low percentage
- fill with modal values
- fill with a third type as other

For this analysis the unique values in the Variable include Self Employed and Salaried. There may be applicants may be unemployed. So it seems logical to fill the missing values with 'Unemployed'

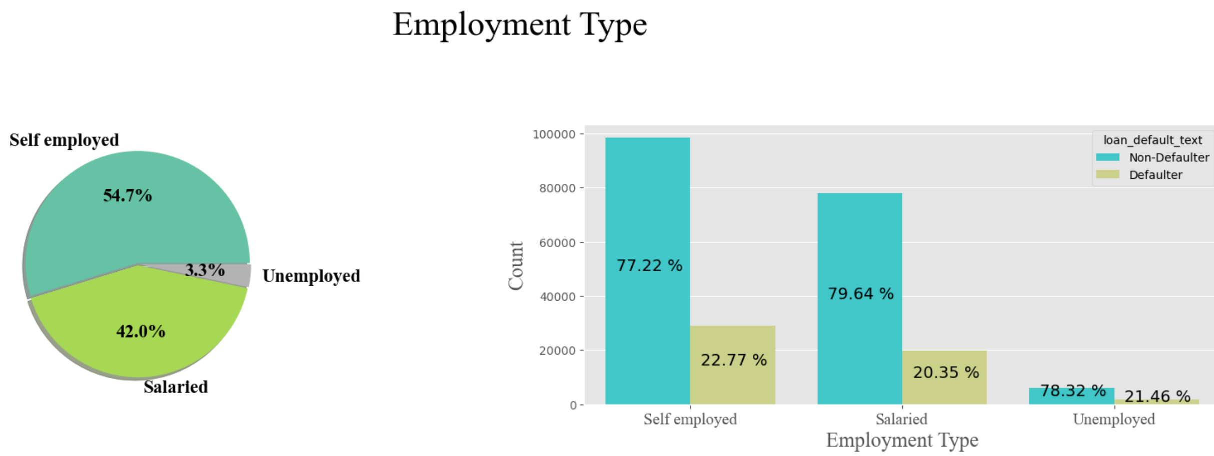
In [28]:

```
data.employment_type.fillna('Unemployed',inplace = True)
```

In [29]:

```
var = 'employment_type'
var_name = var.replace('_',' ').title()
f, axes = plt.subplots(1,2, figsize = (18,6))
vc = data[var].value_counts()
vc.plot.pie(ax = axes[0], radius = 1, cmap = 'Set2' , explode = [0.01,0.01, 0.01], sha
            textprops = {'family': 'Times New Roman','color': 'black','weight': 'bold',
axes[0].set_ylabel('')

sns.countplot(x = var, data = data, hue = 'loan_default_text',ax = axes[1], palette='r
axes[1].set_ylim(0,axes[1].set_ylim()[1]+5)
axes[1].set_xlabel(var_name,fontsize = 18, family = 'Times New Roman')
axes[1].set_ylabel('Count',fontsize = 18, family = 'Times New Roman')
axes[1].set_xticklabels(axes[1].get_xticklabels(),fontsize = 14, family = 'Times New R
vc2 = pd.crosstab(data[var], data.loan_default_text).loc[vc.index]
vc2['Perc_Def'] = (vc2.Defaulter/vc2.sum(axis = 1)*100).round(2)
vc2['Perc_NDef'] = (vc2['Non-Defaulter']/vc2.sum(axis = 1)*100).round(2)
for i in range(len(vc2.index)):
    axes[1].annotate(str(vc2.iloc[i]['Perc_NDef'])+' %', (i - 0.35,vc2.iloc[i]['Non-De
    axes[1].annotate(str(vc2.iloc[i]['Perc_Def'])+' %', (i + 0.05,vc2.iloc[i]['Default
f.suptitle(var_name , fontsize = 30, family = 'Times New Roman')
plt.tight_layout(pad = 4)
plt.show()
```



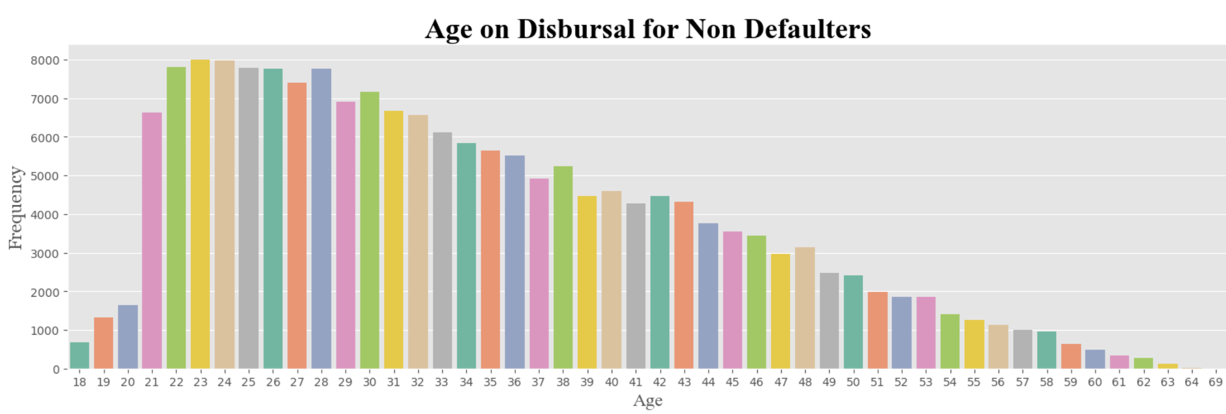
Has age got something to do with defaulting ? what is the distribution of age w.r.t. to defaulters and non-defaulters?

Age in years as on disbursal date

In [30]:
```
data['disbursaldate'] = pd.to_datetime(data['disbursaldate'])
data['date_of_birth'] = pd.to_datetime(data['date_of_birth'])
```

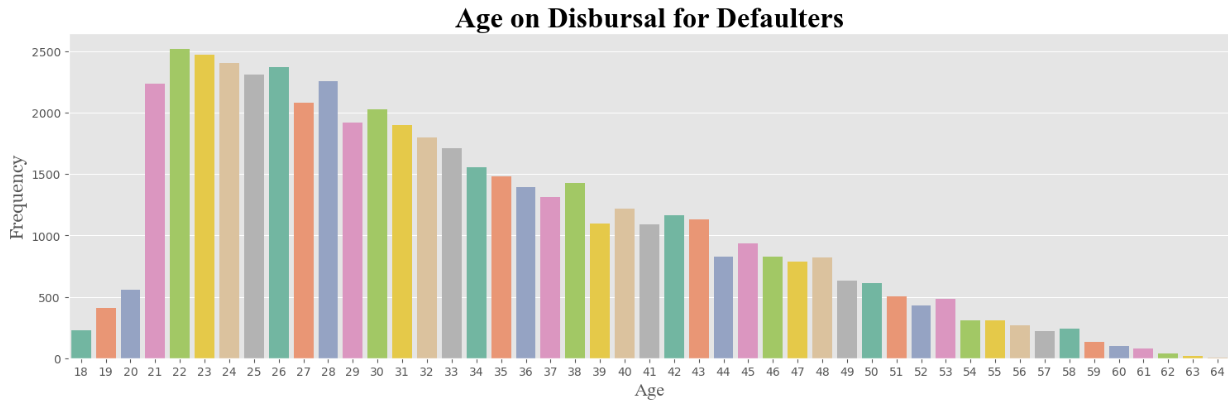
In [31]:
```
data['age_on_disbursal'] = data.apply(lambda row: (row['disbursaldate'] - row['date_of
```

In [32]:
```
plt.figure(figsize = (15,5))
sns.countplot(x='age_on_disbursal', data = data[data.loan_default ==0],palette='Set2')
plt.title('Age on Disbursal for Non Defaulters',family='Times New Roman', weight ='bol
plt.tight_layout( )
plt.xlabel('Age',family='Times New Roman',fontsize= 16)
plt.ylabel('Frequency',family='georgia',fontsize= 16)
plt.show()
```



In [33]:
```
plt.figure(figsize = (15,5))
sns.countplot(x='age_on_disbursal', data = data[data.loan_default ==1],palette='Set2')
plt.title('Age on Disbursal for Defaulters',family='Times New Roman', weight ='bold',f
plt.tight_layout( )
plt.xlabel('Age',family='Times New Roman',fontsize= 16)
```

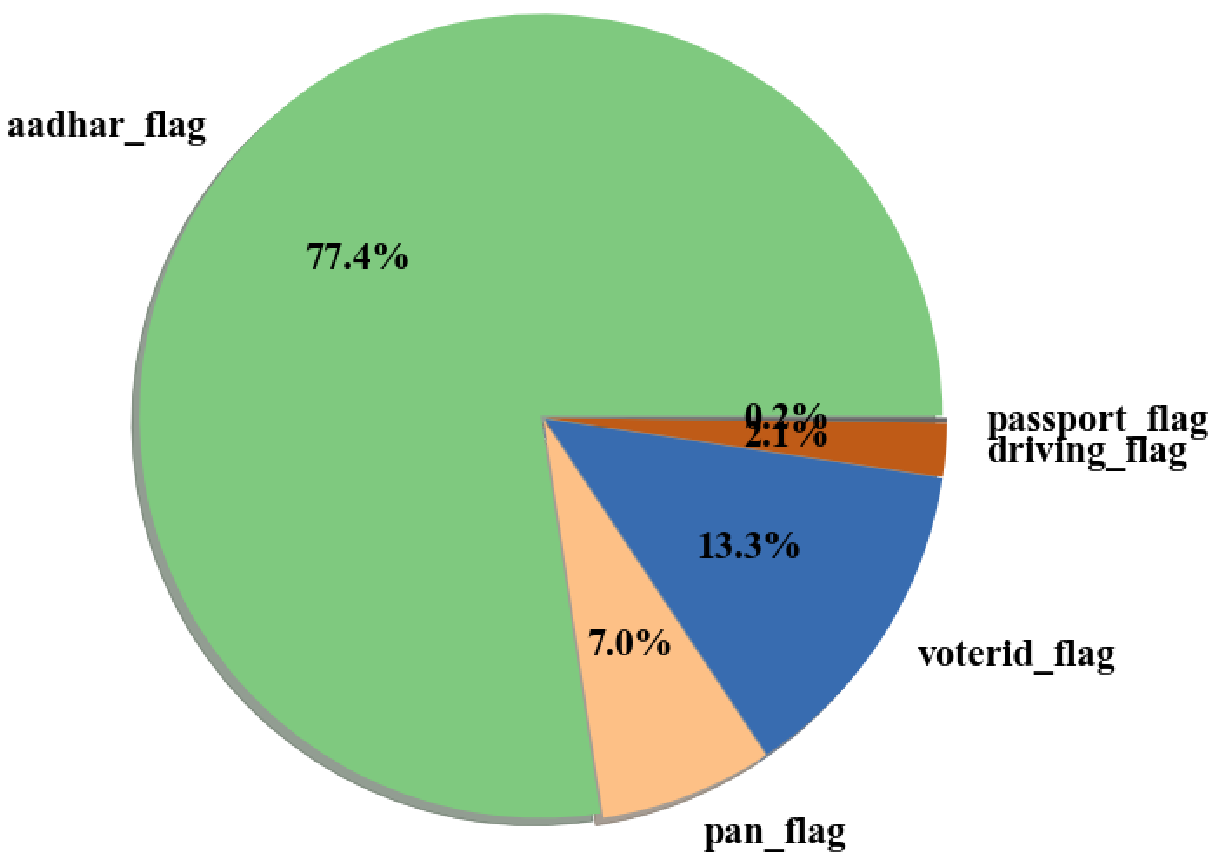
```
plt.ylabel('Frequency',family='georgia',fontsize= 16)
plt.show()
```



What id type was presented by most of the customers as proofs?

In [34]:
```
counts = data[['aadhar_flag', 'pan_flag', 'voterid_flag',
       'driving_flag', 'passport_flag']].sum()

plt.figure(figsize =(20,5))
counts.plot.pie(cmap ='Accent', autopct = '%0.1f%%', radius = 1.5, explode = [0.01]*le
                      textprops = {'family': 'Times New Roman','color': 'black','weight'
plt.ylabel('')
plt.show()
```



a. Study the credit bureau score distribution. How is the distribution for defaulters vs non defaulters? Explore in detail.¶

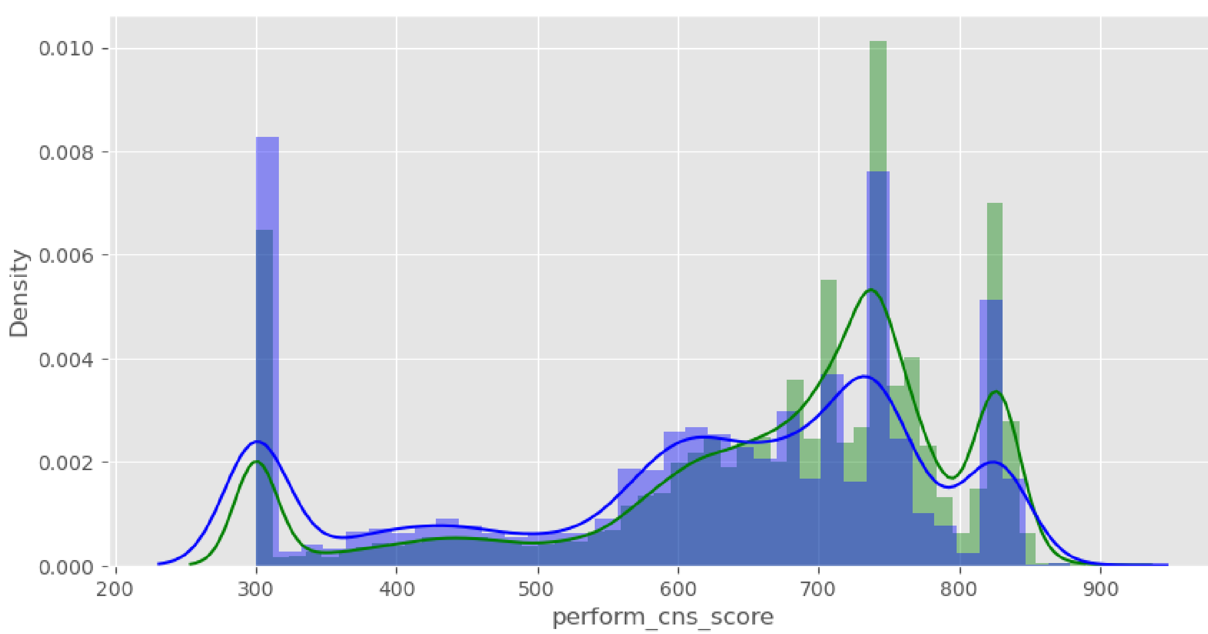
In [35]:
```
data[['perform_cns_score', 'perform_cns_score_description']]
```

Out[35]:

	perform_cns_score	perform_cns_score_description
0	0	No Bureau History Available
1	0	No Bureau History Available
2	0	No Bureau History Available
3	0	No Bureau History Available
4	0	No Bureau History Available
...	...	...
233149	14	Not Scored: Only a Guarantor
233150	14	Not Scored: Only a Guarantor
233151	11	Not Scored: More than 50 active Accounts found
233152	11	Not Scored: More than 50 active Accounts found
233153	11	Not Scored: More than 50 active Accounts found

233154 rows × 2 columns

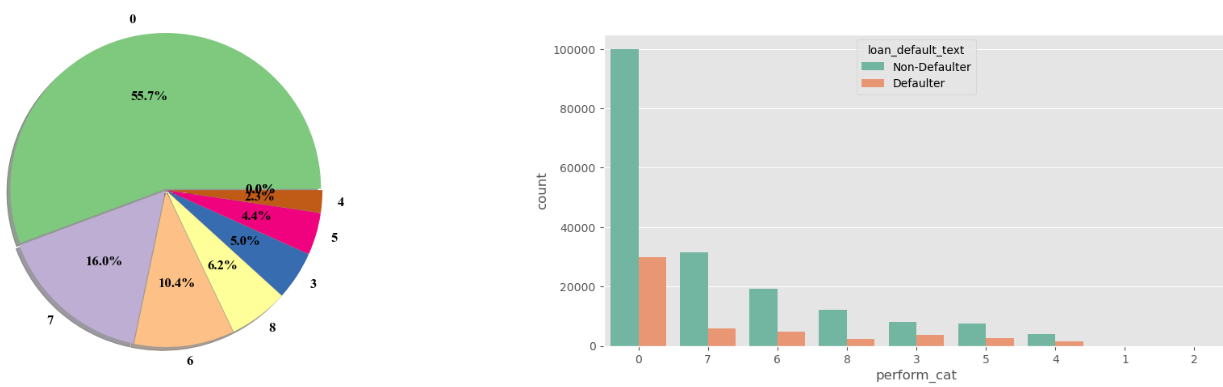
In [36]:
```
plt.figure(figsize = (10,5))
sns.distplot(data[(data.loan_default == 0)& (data.perform_cns_score >=100)].perform_cr
sns.distplot(data[(data.loan_default == 1) & (data.perform_cns_score >=100)].perform_c
plt.show()
```



For both the defaluters and non Defaulters the distribution of CNS score follow a similar distribution

In [37]:
```
data['perform_cat']= pd.cut(data.perform_cns_score,
        bins = range(-1,901,100),
       labels = [0,1,2,3,4,5,6,7,8])
```

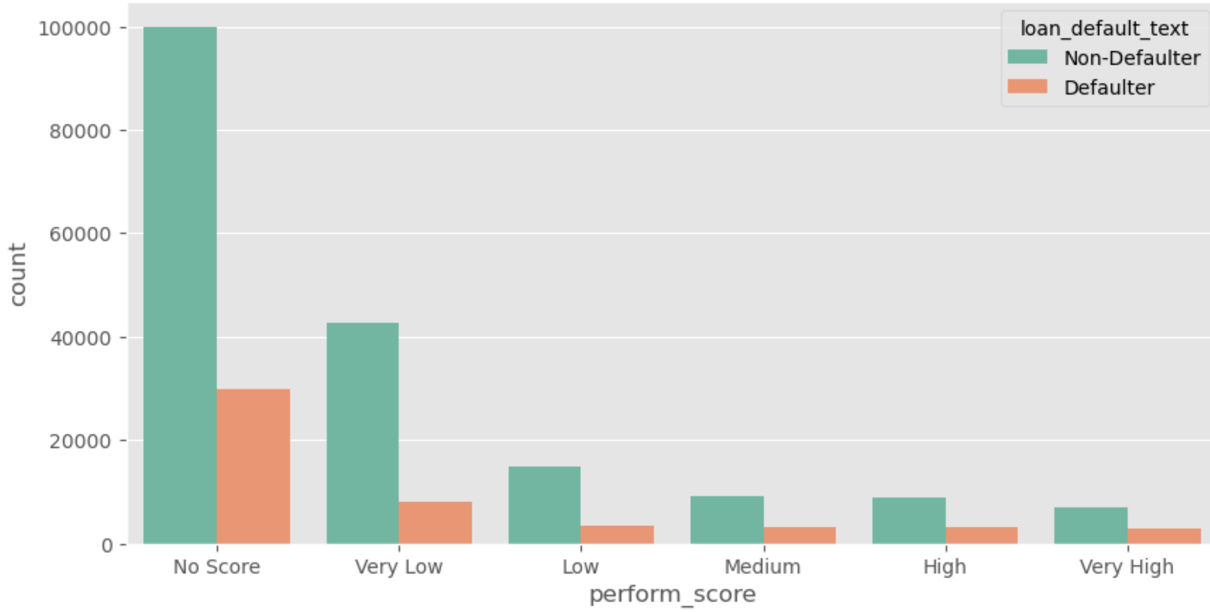
In [38]:
```
f,ax = plt.subplots(1,2, figsize = (22,5))
vc = data.perform_cat.value_counts()
vc.plot.pie(cmap ='Accent', autopct = '%0.1f%%', radius = 1.25, explode = [0.01]*len(v
                    textprops = {'family': 'Times New Roman','color': 'black','weight'
ax[0].set_ylabel('')
sns.countplot(x='perform_cat', hue = 'loan_default_text', data=data,order = vc.index,
plt.show()
```



re categorising the variable

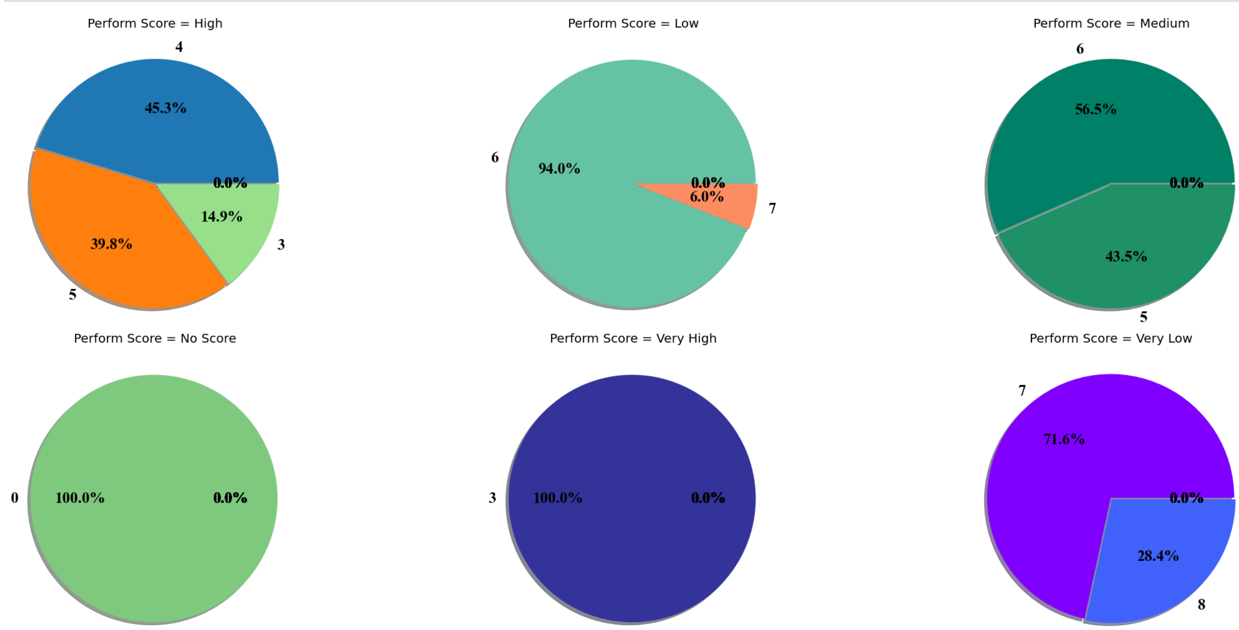
In [39]:
```
data["perform_score"] = 'No Score'
data.loc[data.perform_cns_score_description.str.contains('High'),'perform_score'] = "H
data.loc[data.perform_cns_score_description.str.contains('Very High'),'perform_score']
data.loc[data.perform_cns_score_description.str.contains('Low'),'perform_score'] = "Lo
data.loc[data.perform_cns_score_description.str.contains('Very Low'),'perform_score']
data.loc[data.perform_cns_score_description.str.contains('Medium'),'perform_score'] =
```

In [40]:
```
plt.figure(figsize=(10,5))
sns.countplot(x='perform_score',hue='loan_default_text',data=data, palette='Set2')
plt.show()
```



In [41]:

```
cmap = ['tab20','Set2','summer','Accent','terrain', 'rainbow','Paired']
perf_cat = list(data.perform_score.unique())
perf_cat.sort()
f,ax = plt.subplots(2,int(data.perform_score.nunique()/2),figsize = (25,10))
k = 0
for j in range(2):
    for i in range(int(len(perf_cat)/2)):
        subdata = data[data.perform_score==perf_cat[k]].copy()
        vc = subdata.perform_cat.value_counts()
        vc.plot.pie(cmap =cmap[k], autopct = '%0.1f%%', radius = 1.25, explode = [0.01
                        textprops = {'family': 'Times New Roman','color': 'black','wei
        ax[j,i].set_ylabel('')
        ax[j,i].set_title('Perform Score = '+str(perf_cat[k])+'\n\n')
        k += 1
plt.tight_layout(w_pad = 1)
```



Explore the primary and secondary account details. Is the information in some way related to loan default probability ?

In [42]:
```
pri_account_info = data.columns[data.columns.str.contains('pri')]
```

exploring the primary account details¶

In [43]:
```
i = 1
for col in pri_account_info:
    f, (ax_box, ax_hist) = plt.subplots(2,1, figsize = (12,5),sharex = True, gridspec_
    plt.suptitle('Figure' + str(i) + ': Histogram and Bos Plot of ' + col.replace('_',
    sns.boxplot(data[col], ax= ax_box,color = 'red')
    sns.distplot(data[col], ax = ax_hist, color = 'magenta')
    sns.despine(ax = ax_box, left = True)
    sns.despine(ax= ax_hist, left = True)
    i = i+1
    plt.show()
```

Figure1: Histogram and Bos Plot of Pri No Of Accts

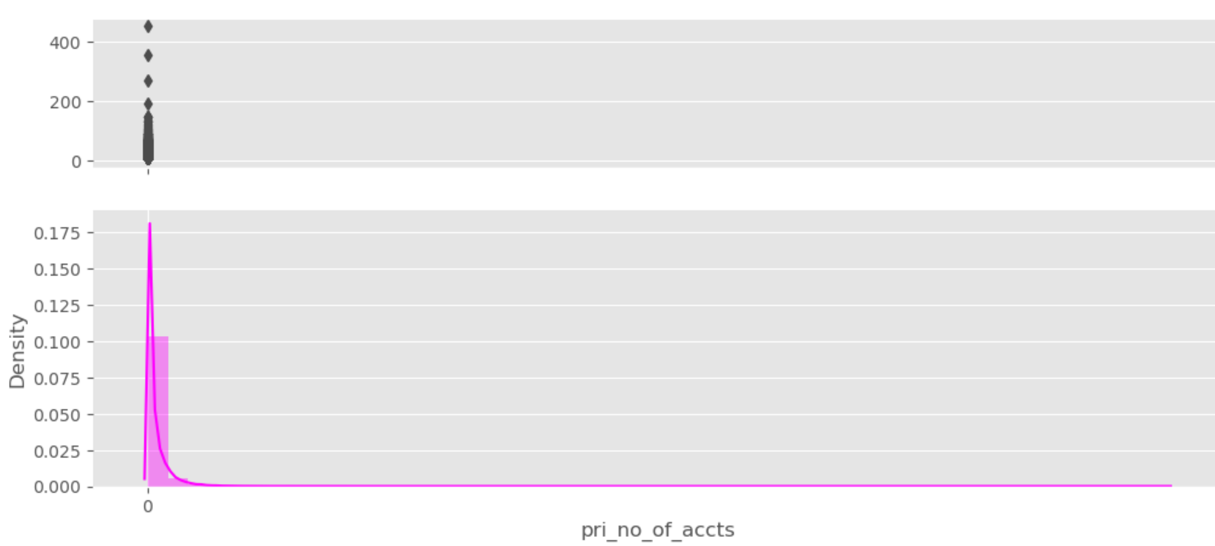


Figure2: Histogram and Bos Plot of Pri Active Accts

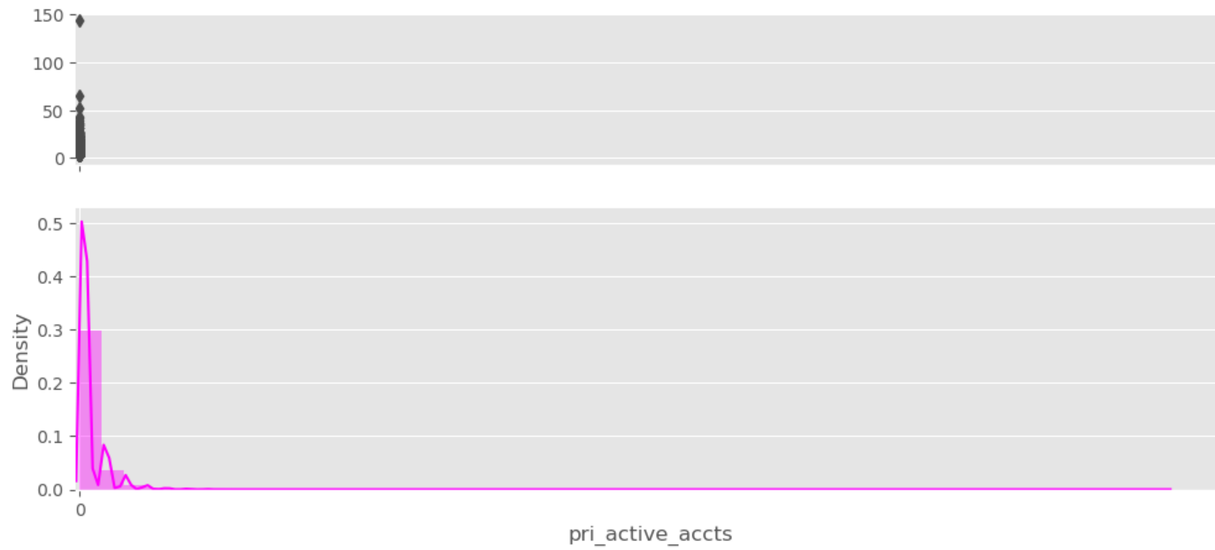


Figure3: Histogram and Bos Plot of Pri Overdue Accts

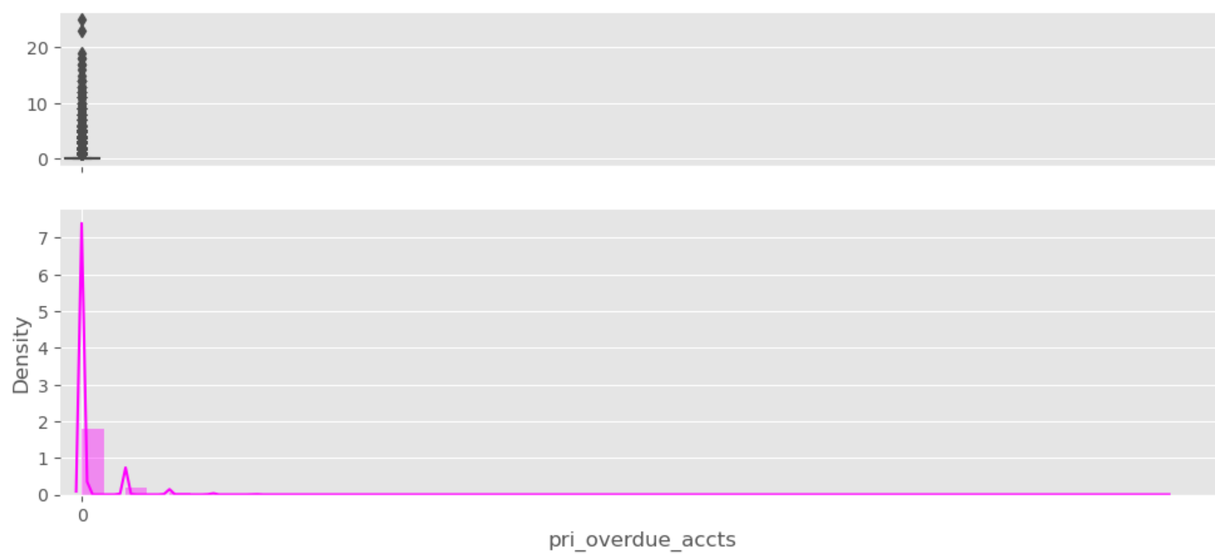


Figure4: Histogram and Bos Plot of Pri Current Balance

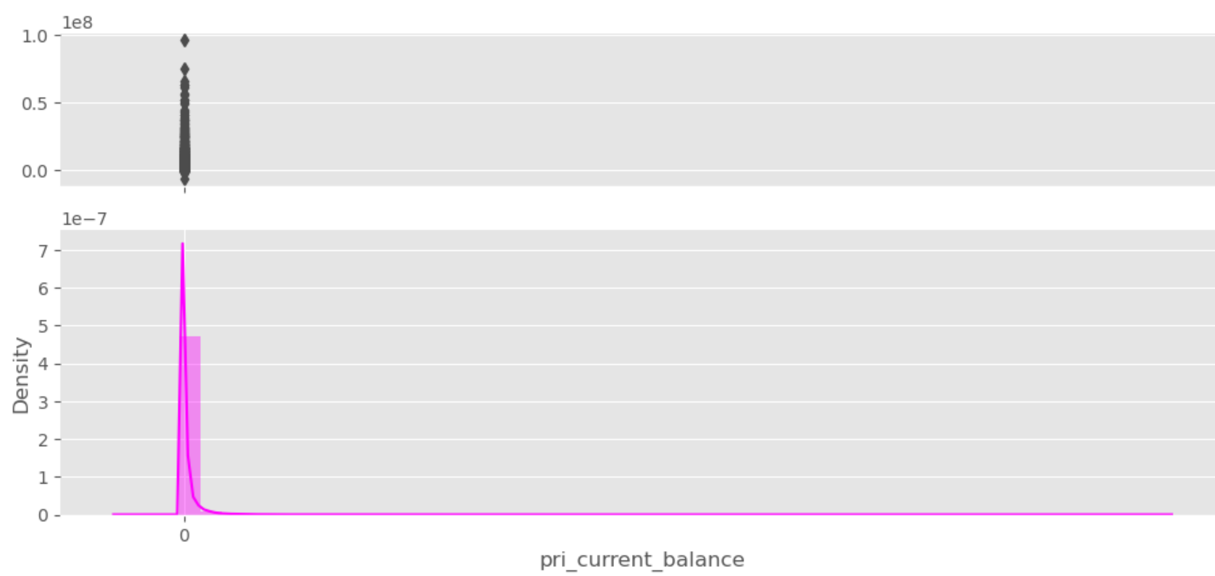


Figure5: Histogram and Bos Plot of Pri Sanctioned Amount

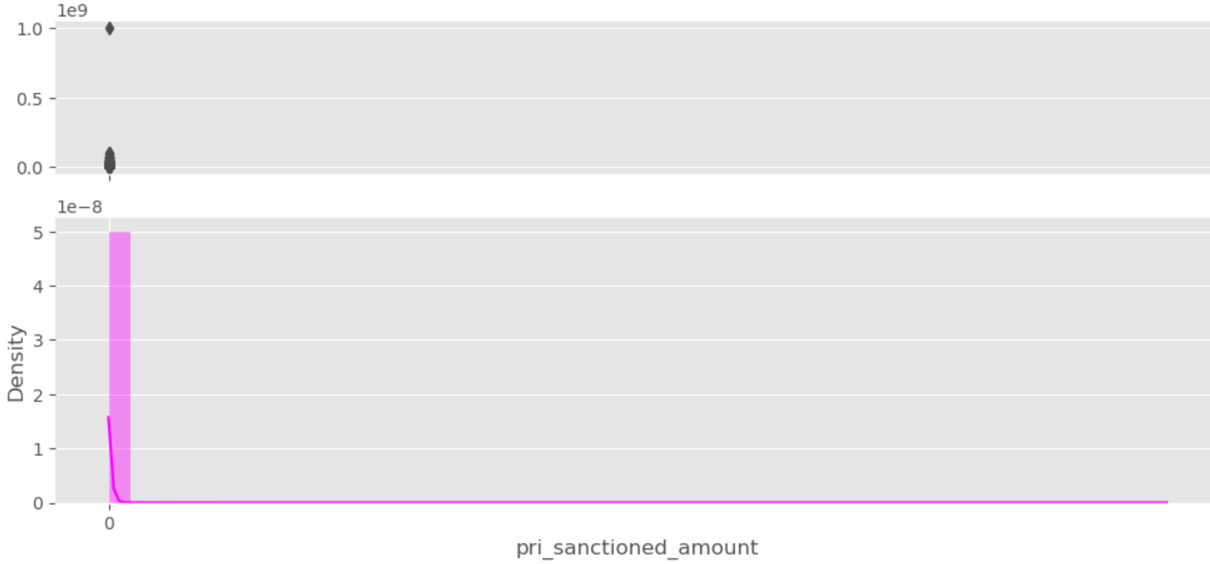


Figure6: Histogram and Bos Plot of Pri Disbursed Amount

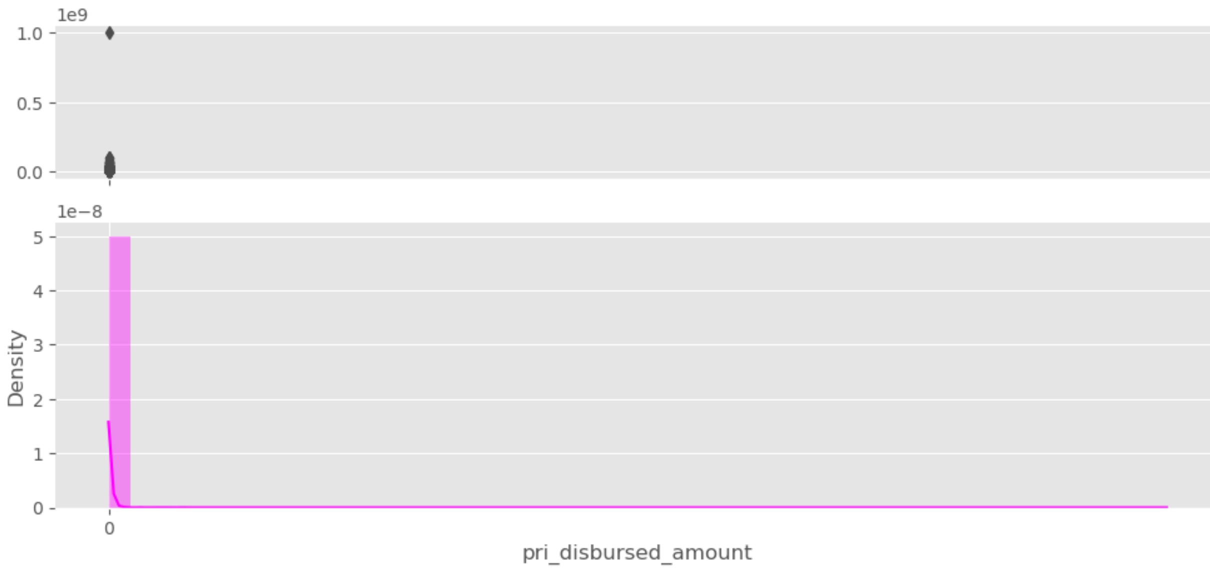
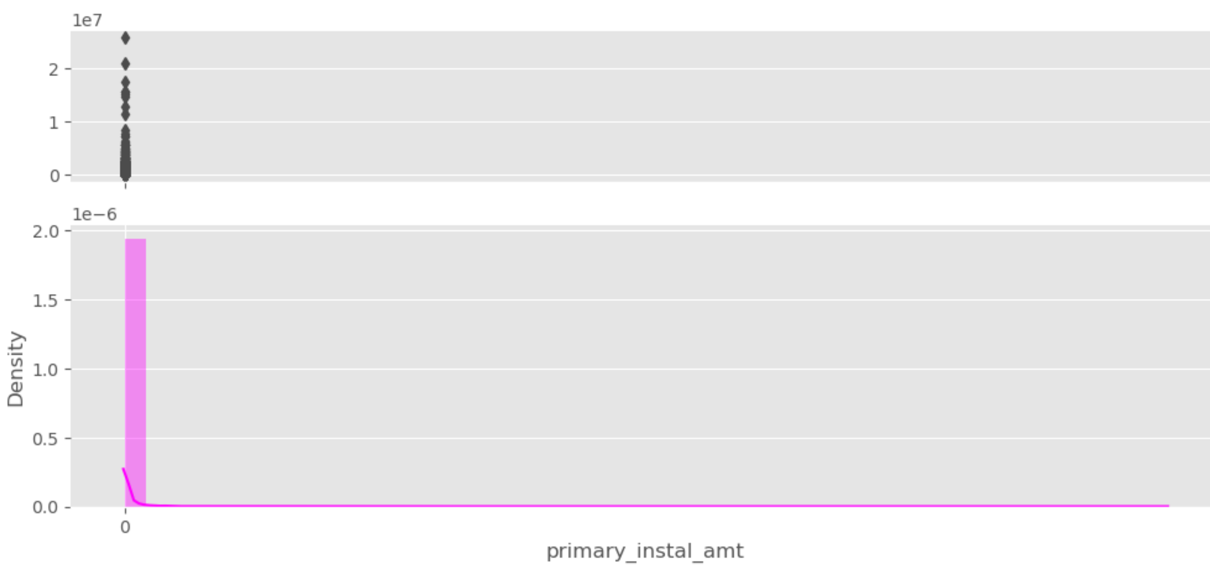


Figure7: Histogram and Bos Plot of Primary Instal Amt



Exploring Secondary account information

In [44]:
```
sec_account_info = data.columns[data.columns.str.contains('sec')]
```

In [45]:
```
i = 1
for col in sec_account_info:
    f, (ax_box, ax_hist) = plt.subplots(2,1, figsize = (12,5),sharex = True, gridspec_
    plt.suptitle('Figure' + str(i) + ': Histogram and Bos Plot of ' + col.replace('_',
    sns.boxplot(data[col], ax= ax_box,color = 'red')
    sns.distplot(data[col], ax = ax_hist, color = 'magenta')
    sns.despine(ax = ax_box, left = True)
    sns.despine(ax= ax_hist, left = True)
    i = i+1
    plt.show()
```

Figure1: Histogram and Bos Plot of Sec No Of Accts

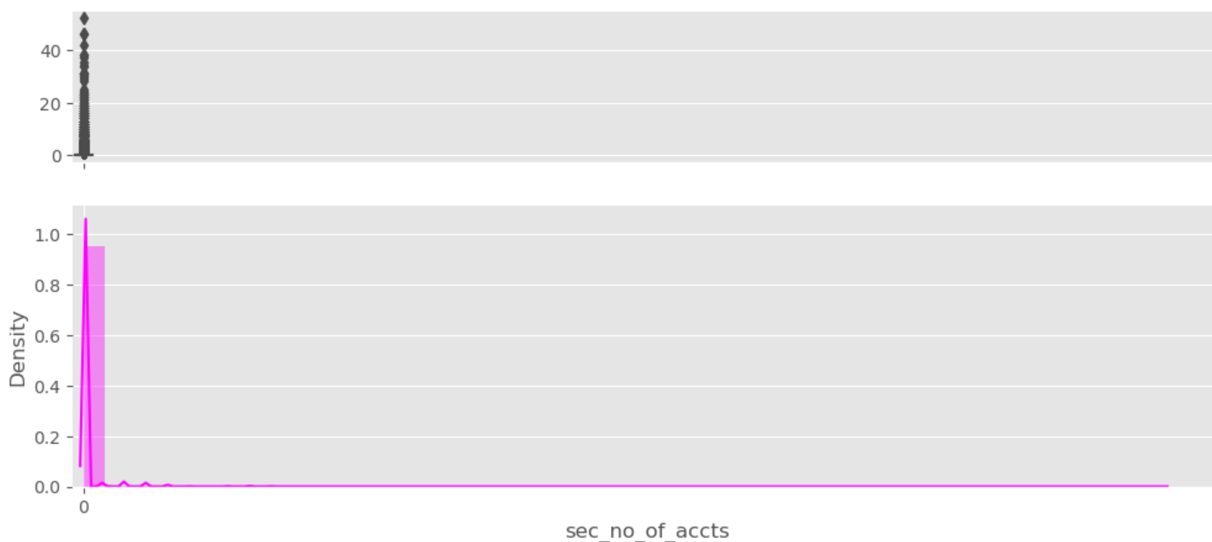


Figure2: Histogram and Bos Plot of Sec Active Accts

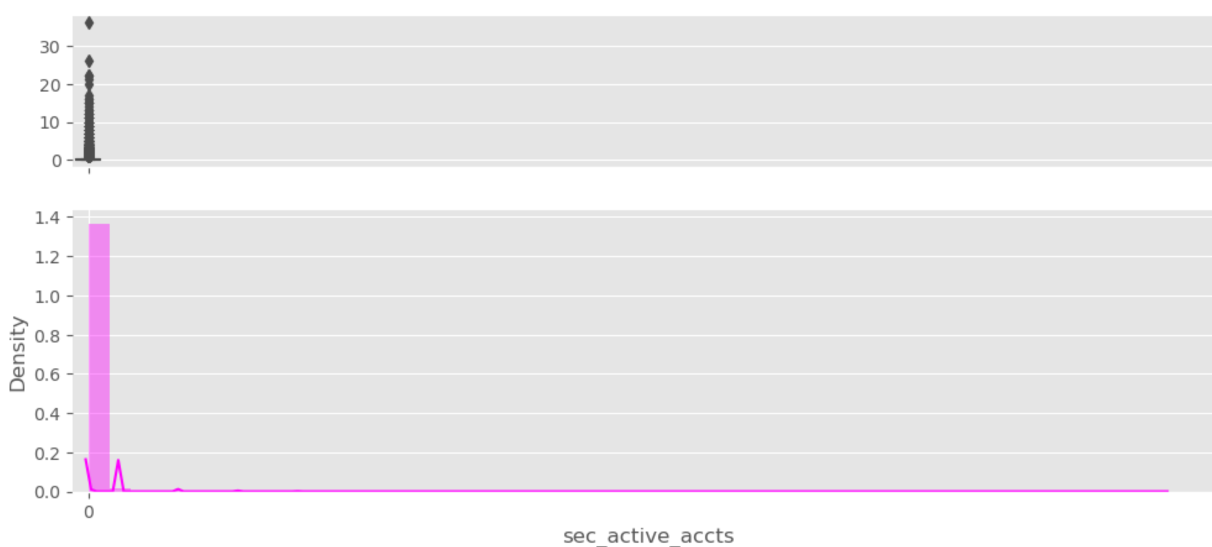


Figure3: Histogram and Bos Plot of Sec Overdue Accts

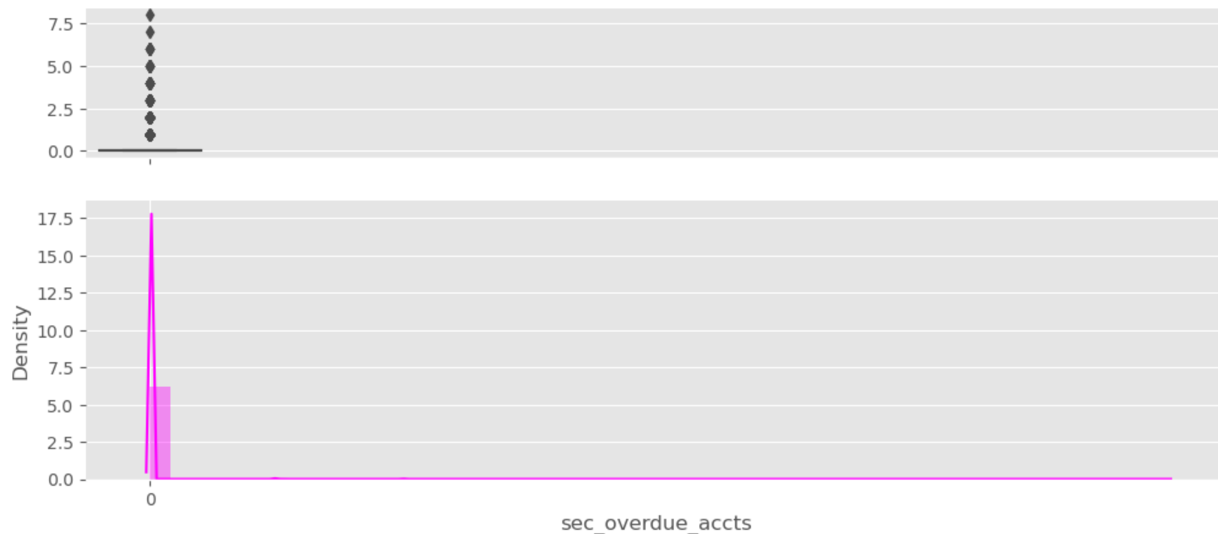


Figure4: Histogram and Bos Plot of Sec Current Balance

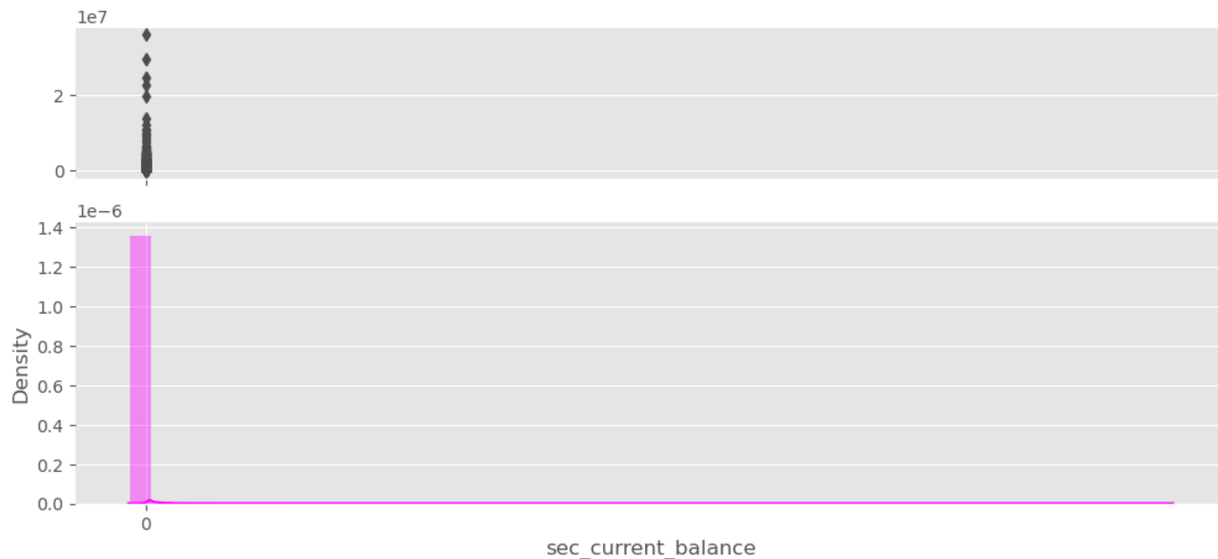


Figure5: Histogram and Bos Plot of Sec Sanctioned Amount

Figure6: Histogram and Bos Plot of Sec Disbursed Amount

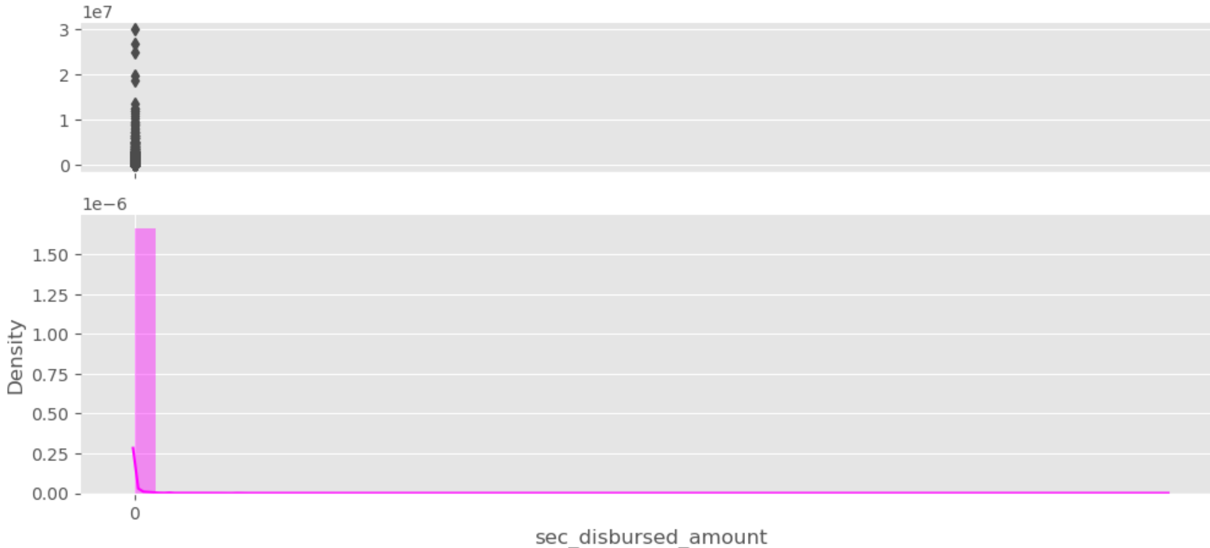
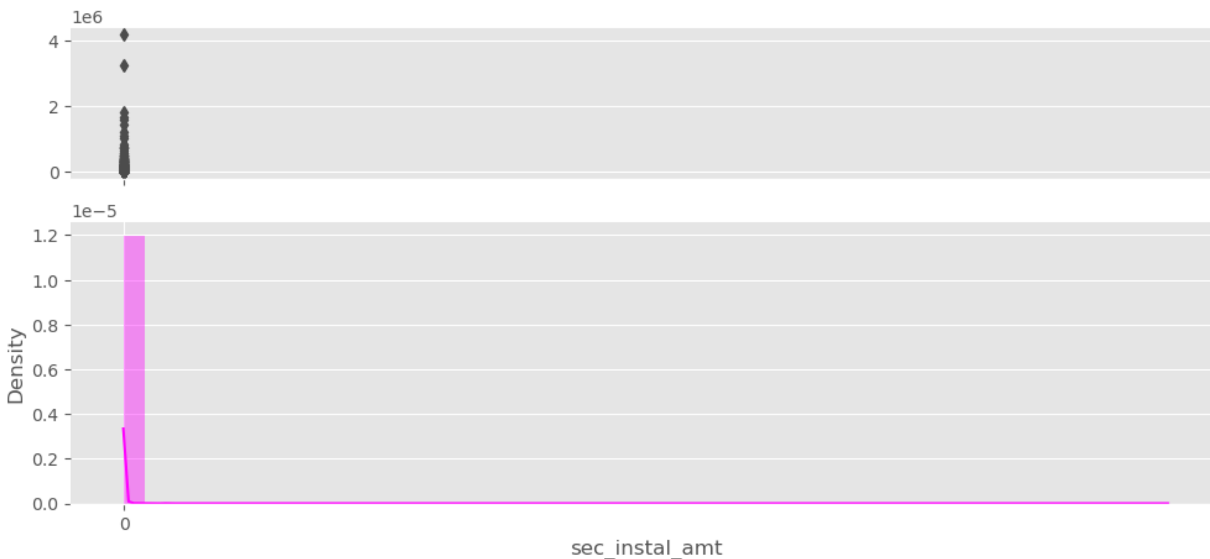


Figure7: Histogram and Bos Plot of Sec Instal Amt



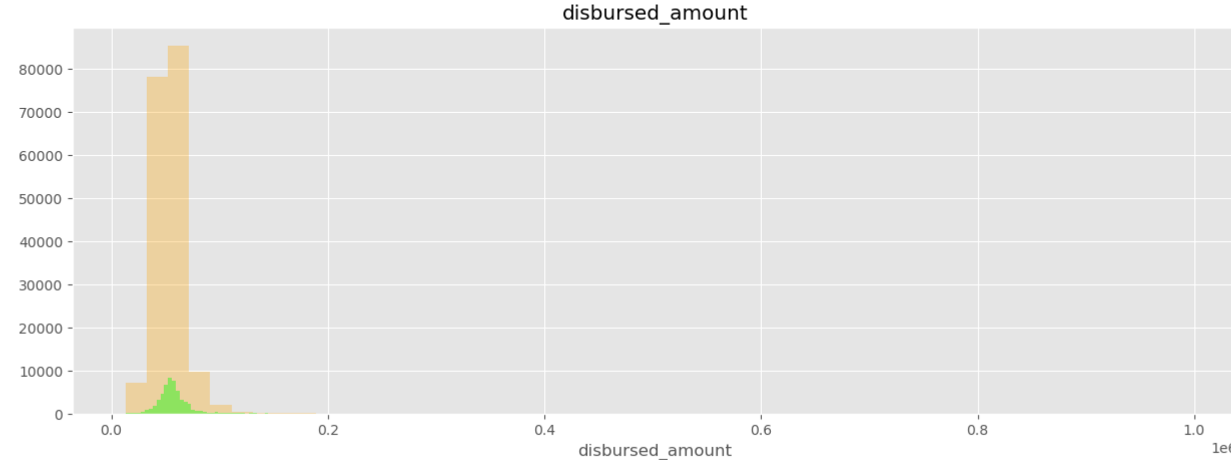
Explore the distribution of other variables w.r.t. target

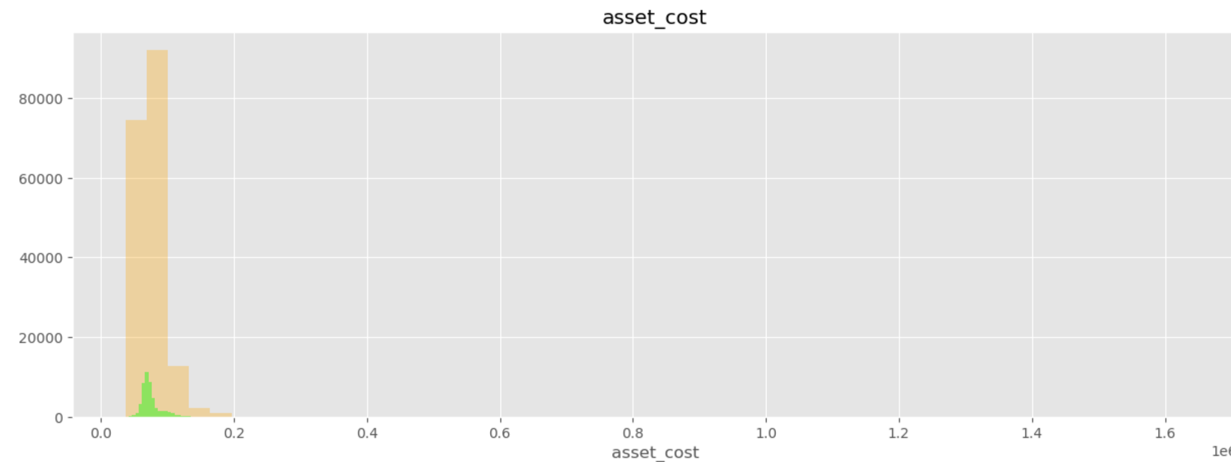
cols to be used

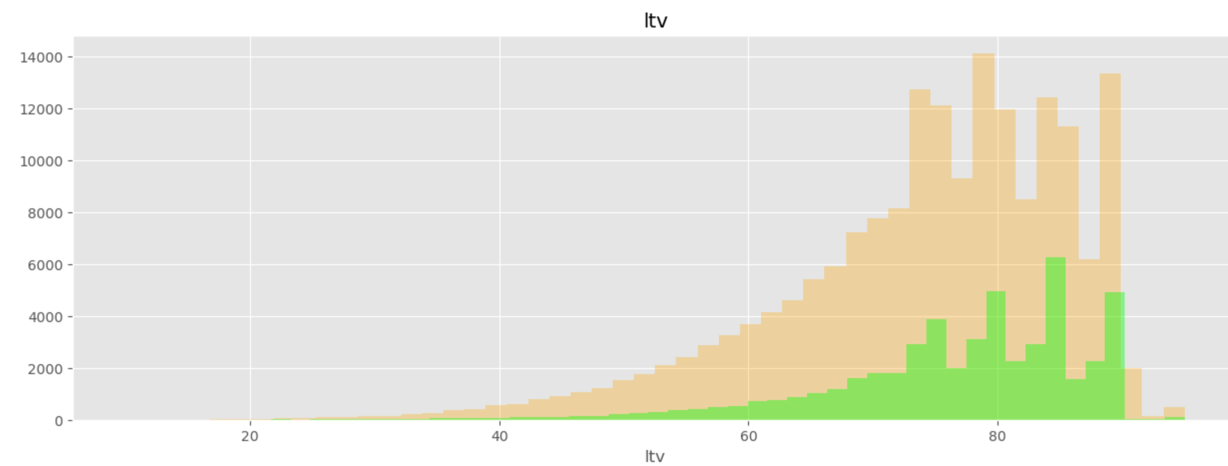
In [46]:
```
cols = ['disbursed_amount', 'asset_cost', 'ltv','perform_cns_score','pri_current_balar
        'sec_current_balance', 'sec_sanctioned_amount','sec_disbursed_amount','primary
```

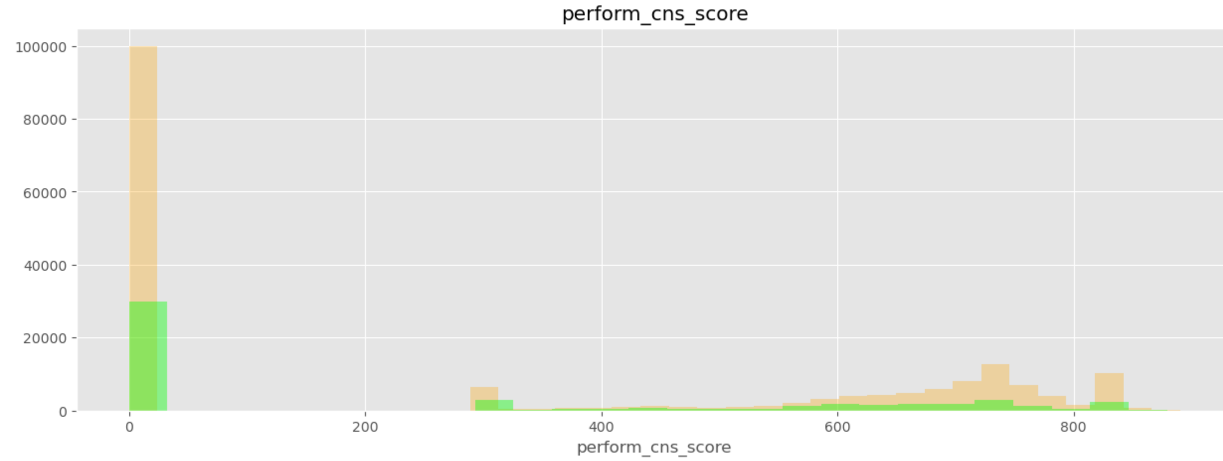
In [47]:
```
for i in cols:
    plt.figure(figsize = (15,5))
    data1 = data.loc[data.loan_default==0, i]
    data2 = data.loc[data.loan_default==1, i]
    sns.distplot(data1, kde = False, color= 'orange', hist_kws = {'alpha' : 0.3})
    sns.distplot(data2, kde = False, color = 'lime')
```

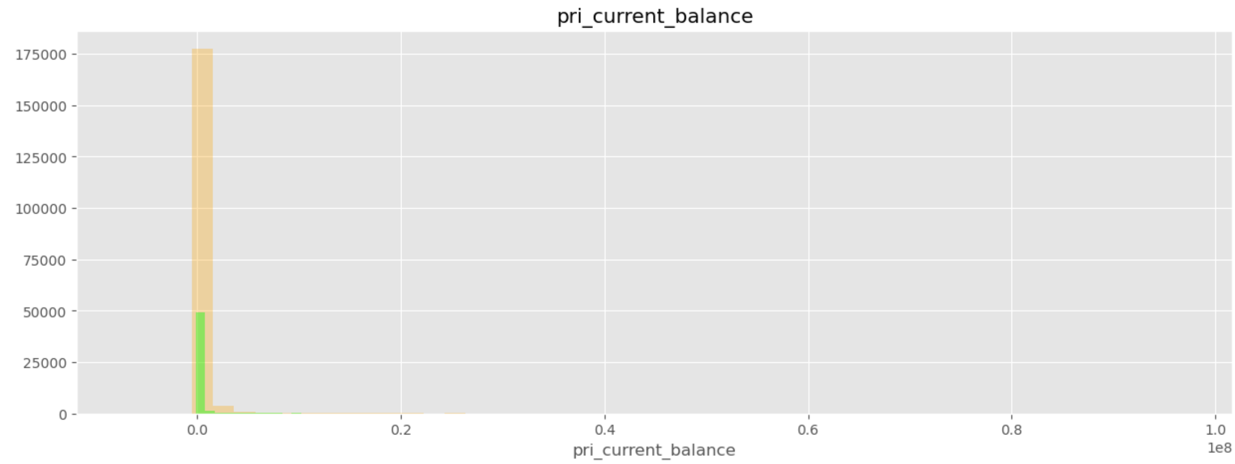
```
    plt.title(i)
    plt.show()
```

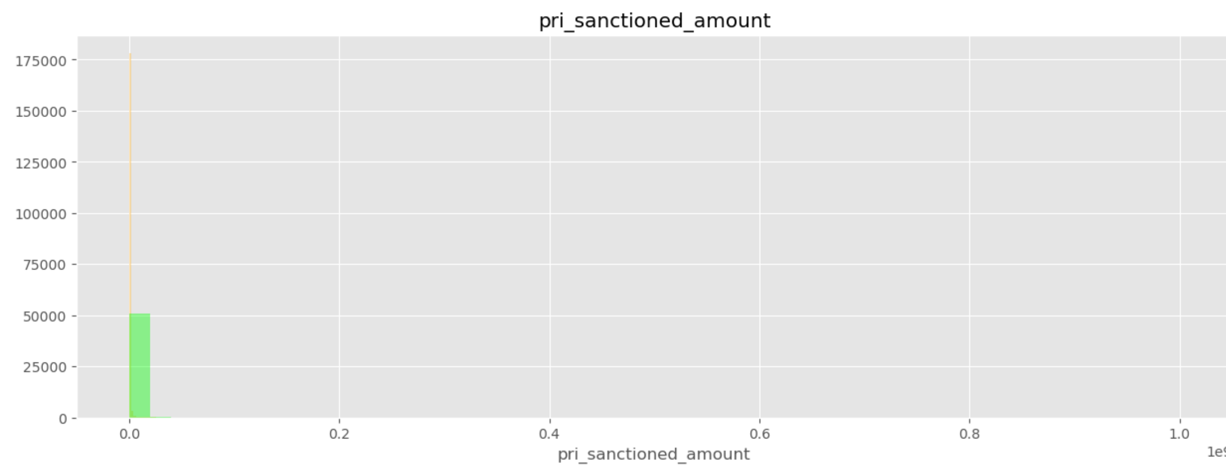


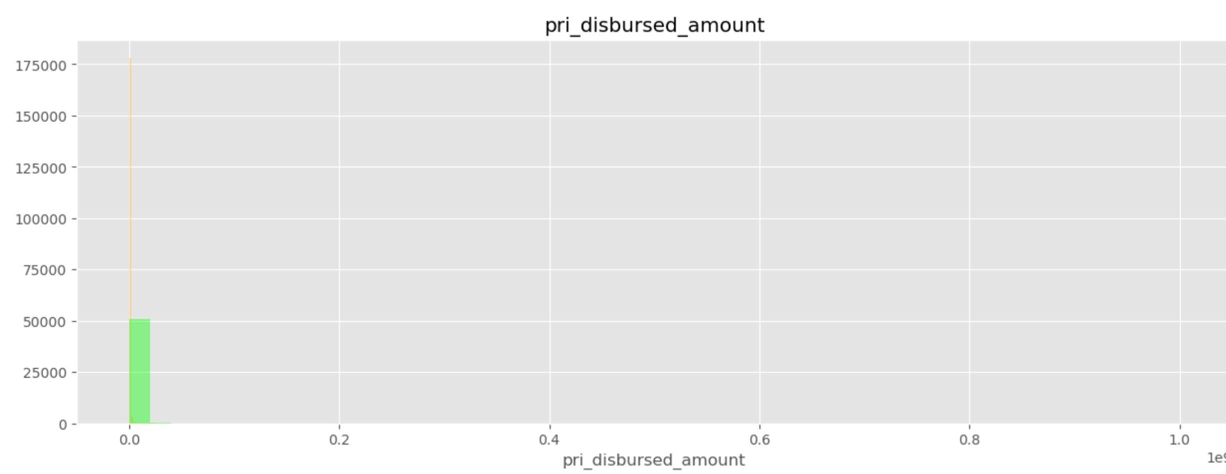


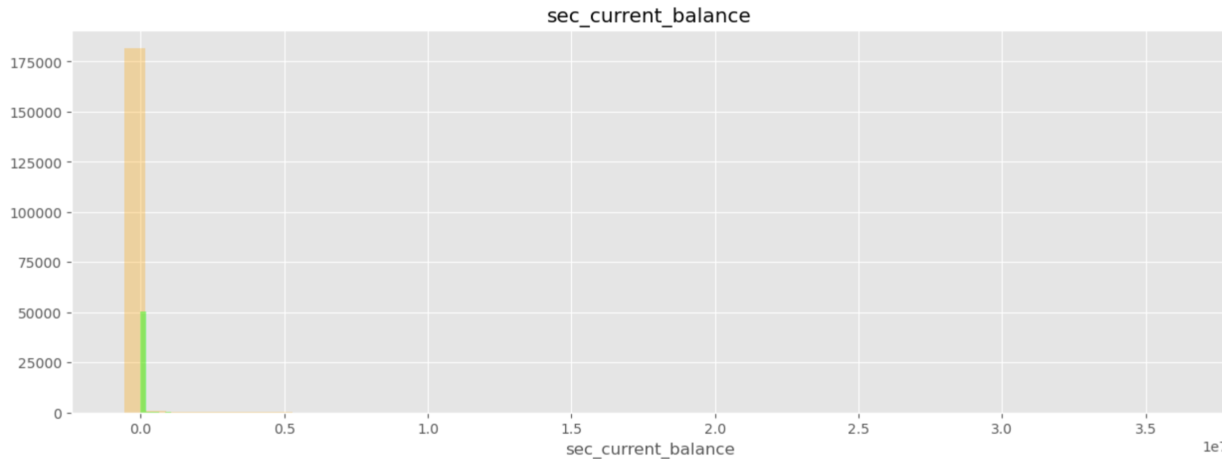


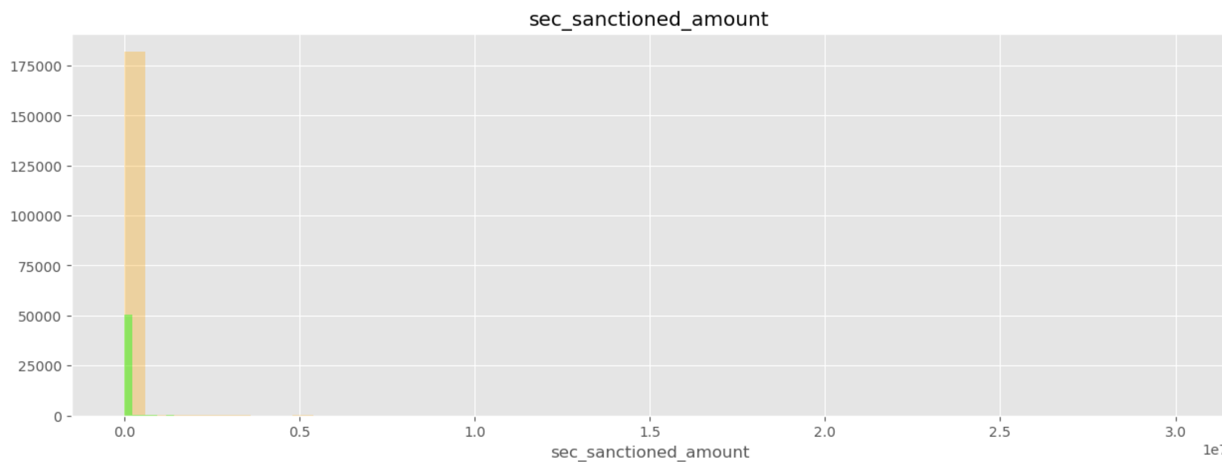

perform_cns_score
100000
80000
60000
40000
20000
0
0
200
400
600
800
perform_cns_score

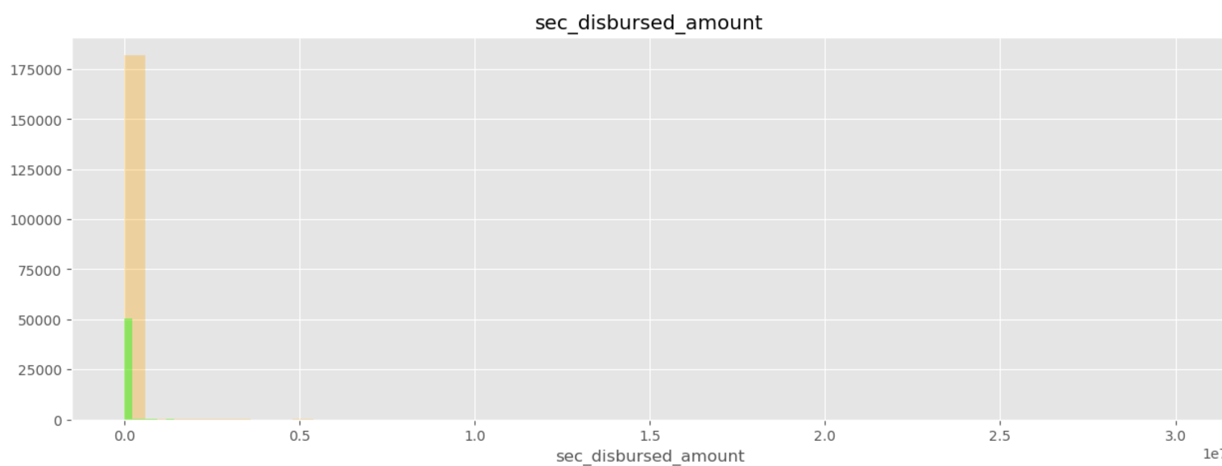

pri_current_balance
175000
150000
125000
100000
75000
50000
25000
0
0.0
0.2
0.4
0.6
0.8
1.0
1e8
pri_current_balance

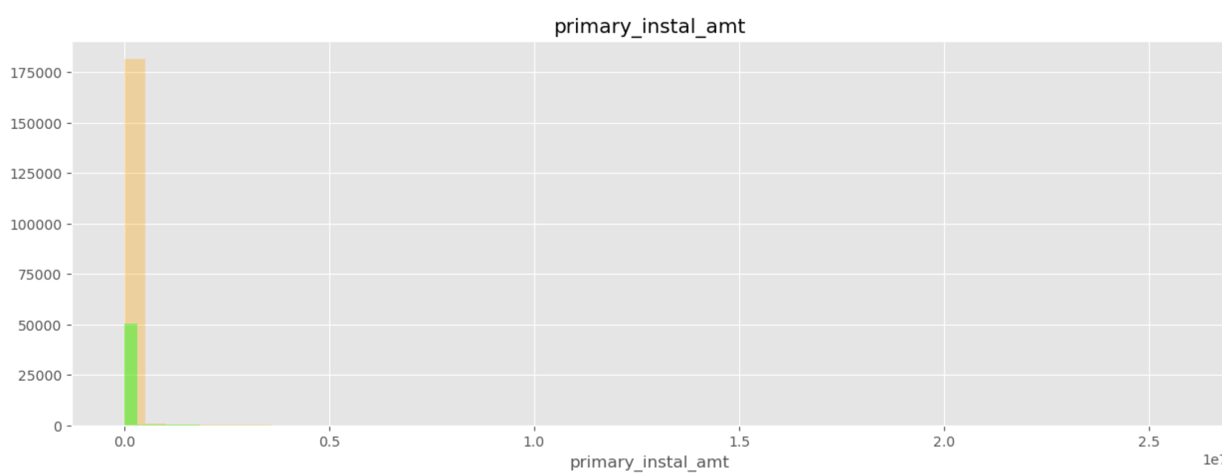

pri_sanctioned_amount
175000
150000
125000
100000
75000
50000
25000
0
0.0
0.2
0.4
0.6
0.8
1.0
1e9
pri_sanctioned_amount

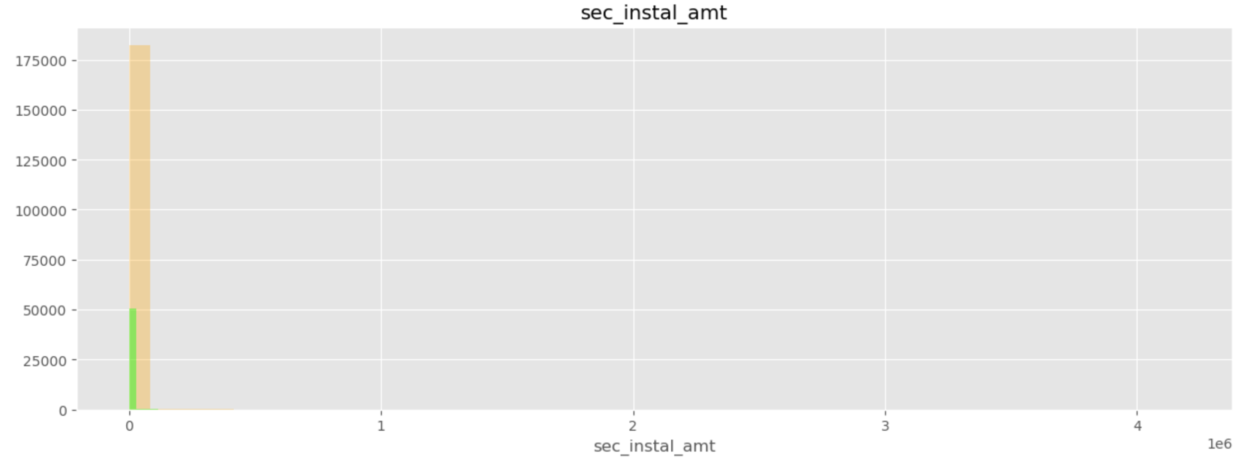

pri_disbursed_amount
175000
150000
125000
100000
75000
50000
25000
0
0.0
0.2
0.4
0.6
0.8
1.0
1e9
pri_disbursed_amount


sec_current_balance
175000
150000
125000
100000
75000
50000
25000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
1e7
sec_current_balance


sec_sanctioned_amount
175000
150000
125000
100000
75000
50000
25000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1e7
sec_sanctioned_amount


sec_disbursed_amount
175000
150000
125000
100000
75000
50000
25000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1e7
sec_disbursed_amount


primary_instal_amt
175000
150000
125000
100000
75000
50000
25000
0
0.0
0.5
1.0
1.5
2.0
2.5
1e7
primary_instal_amt



perform a baseline predictive analytics.

selecting features to drop

In [48]:
```
a = (list(data.columns))
```

In [49]:
```
a.sort()
```

In [50]:
```
def text_months(x):
    year = int(re.findall('\d',x)[0])
    month = int(re.findall('\d',x)[1])
    total = year*12 +month
    return total
```

In [ ]:
```
# Creating new categories
```

In [51]:
```
data['avg_acnt_age_month'] = data.average_acct_age.astype('str').apply(text_months)
data['credit_history_months'] = data.credit_history_length.astype('str').apply(text_mo
```

In [52]:
```
data["credit_hist_cat"] = "Low"
data.loc[data.credit_history_months <= 48,"credit_hist_cat"] = "Low"
data.loc[data.credit_history_months > 48,"credit_hist_cat"] = "Medium"
data.loc[data.credit_history_months> 96,"credit_hist_cat"] = "High"
```

In [53]:
```
data["loan_tenure"] = 'least_pref'
data.loc[data.avg_acnt_age_month<= 60,"loan_tenure"] ="most_pref"
```

In [54]:
```
data["preferred_age"] = 'Negative'
data.loc[data.age_on_disbursal > 100,  "preferred_age"] = "least_pref"
data.loc[data.age_on_disbursal >= 336, "preferred_age"] = "most_pref"
```

In [55]:
```
cat_columns = ['branch_id', 'supplier_id', 'manufacturer_id', 'state_id', 'employment_
                'mobileno_avl_flag', 'aadhar_flag', 'pan_flag', 'voterid_flag', 'drivir
                'new_accts_in_last_six_months','delinquent_accts_in_last_six_months',
                'pri_no_of_accts', 'pri_active_accts','pri_overdue_accts','sec_no_of_ac
                 'perform_cat', 'perform_score','preferred_age','loan_default']
```

```
quant_columns = ['disbursed_amount', 'asset_cost', 'ltv','pri_current_balance', 'pri_s
                 'sec_current_balance', 'sec_sanctioned_amount', 'sec_disbursed_amount'
```

In [56]:
```
final_columns = cat_columns + quant_columns
```

In [57]:
```
cat_columns
```

Out[57]:
```
['branch_id',
 'supplier_id',
 'manufacturer_id',
 'state_id',
 'employment_type',
 'mobileno_avl_flag',
 'aadhar_flag',
 'pan_flag',
 'voterid_flag',
 'driving_flag',
 'passport_flag',
 'new_accts_in_last_six_months',
 'delinquent_accts_in_last_six_months',
 'pri_no_of_accts',
 'pri_active_accts',
 'pri_overdue_accts',
 'sec_no_of_accts',
 'sec_active_accts',
 'sec_overdue_accts',
 'loan_tenure',
 'credit_hist_cat',
 'perform_cat',
 'perform_score',
 'preferred_age',
 'loan_default']
```

In [58]:
```
final_columns
```

Out[58]:
```
['branch_id',
 'supplier_id',
 'manufacturer_id',
 'state_id',
 'employment_type',
 'mobileno_avl_flag',
 'aadhar_flag',
 'pan_flag',
 'voterid_flag',
 'driving_flag',
 'passport_flag',
 'new_accts_in_last_six_months',
 'delinquent_accts_in_last_six_months',
 'pri_no_of_accts',
 'pri_active_accts',
 'pri_overdue_accts',
 'sec_no_of_accts',
 'sec_active_accts',
 'sec_overdue_accts',
 'loan_tenure',
 'credit_hist_cat',
 'perform_cat',
 'perform_score',
 'preferred_age',
 'loan_default',
 'disbursed_amount',
 'asset_cost',
 'ltv',
 'pri_current_balance',
 'pri_sanctioned_amount',
 'pri_disbursed_amount',
 'sec_current_balance',
 'sec_sanctioned_amount',
 'sec_disbursed_amount',
 'primary_instal_amt',
 'sec_instal_amt']
```

In [59]:
```
final_data = data[final_columns]
```

In [60]:
```
final_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 233154 entries, 0 to 233153
Data columns (total 36 columns):
 #   Column                               Non-Null Count   Dtype   
---  ------                               --------------   -----   
 0   branch_id                            233154 non-null  category
 1   supplier_id                          233154 non-null  category
 2   manufacturer_id                      233154 non-null  category
 3   state_id                             233154 non-null  category
 4   employment_type                      233154 non-null  object  
 5   mobileno_avl_flag                    233154 non-null  int64   
 6   aadhar_flag                          233154 non-null  int64   
 7   pan_flag                             233154 non-null  int64   
 8   voterid_flag                         233154 non-null  int64   
 9   driving_flag                         233154 non-null  int64   
 10  passport_flag                        233154 non-null  int64   
 11  new_accts_in_last_six_months         233154 non-null  category
 12  delinquent_accts_in_last_six_months  233154 non-null  category
 13  pri_no_of_accts                      233154 non-null  category
 14  pri_active_accts                     233154 non-null  category
 15  pri_overdue_accts                    233154 non-null  category
 16  sec_no_of_accts                      233154 non-null  category
 17  sec_active_accts                     233154 non-null  category
 18  sec_overdue_accts                    233154 non-null  category
 19  loan_tenure                          233154 non-null  object  
 20  credit_hist_cat                      233154 non-null  object  
 21  perform_cat                          233154 non-null  category
 22  perform_score                        233154 non-null  object  
 23  preferred_age                        233154 non-null  object  
 24  loan_default                         233154 non-null  int64   
 25  disbursed_amount                     233154 non-null  int64   
 26  asset_cost                           233154 non-null  int64   
 27  ltv                                  233154 non-null  float64 
 28  pri_current_balance                  233154 non-null  int64   
 29  pri_sanctioned_amount                233154 non-null  int64   
 30  pri_disbursed_amount                 233154 non-null  int64   
 31  sec_current_balance                  233154 non-null  int64   
 32  sec_sanctioned_amount                233154 non-null  int64   
 33  sec_disbursed_amount                 233154 non-null  int64   
 34  primary_instal_amt                   233154 non-null  int64   
 35  sec_instal_amt                       233154 non-null  int64   
dtypes: category(13), float64(1), int64(17), object(5)
memory usage: 44.1+ MB
```

In [61]:

```
final_data
```

Out[61]:

	branch_id	supplier_id	manufacturer_id	state_id	employment_type	mobileno_avl_flag	aadha
0	67	22807	45	6	Salaried	1	
1	67	22807	45	6	Self employed	1	
2	67	22807	45	6	Self employed	1	
3	67	22807	45	6	Salaried	1	
4	67	22744	86	6	Self employed	1	
...	...	...	...	...	...	...	
233149	5	22289	51	9	Self employed	1	
233150	138	17408	51	9	Self employed	1	
233151	135	23313	45	4	Self employed	1	
233152	160	16212	48	16	Self employed	1	
233153	3	14573	45	1	Self employed	1	

233154 rows × 36 columns

analysing relation ship using pair plot

In [ ]:

```
sns.pairplot(final_data, hue = 'loan_default', palette='Set1')
```

split into train and test

In [62]:

```
final_data.describe()
```

Out[62]:

	mobileno_avl_flag	aadhar_flag	pan_flag	voterid_flag	driving_flag	passport_flag
count	233154.0	233154.00000	233154.000000	233154.000000	233154.000000	233154.000000
mean	1.0	0.84032	0.075577	0.144943	0.023242	0.002127
std	0.0	0.36631	0.264320	0.352044	0.150672	0.046074
min	1.0	0.00000	0.000000	0.000000	0.000000	0.000000
25%	1.0	1.00000	0.000000	0.000000	0.000000	0.000000
50%	1.0	1.00000	0.000000	0.000000	0.000000	0.000000
75%	1.0	1.00000	0.000000	0.000000	0.000000	0.000000
max	1.0	1.00000	1.000000	1.000000	1.000000	1.000000

In [63]:
```
for i in cat_columns:
    if i != 'loan_default':
        final_data[i]= final_data[i].astype('object')
```

In [64]:
```
final_data.describe()
```

Out[64]:

	loan_default	disbursed_amount	asset_cost	ltv	pri_current_balance	pri_sanctio
count	233154.000000	233154.000000	2.331540e+05	233154.000000	2.331540e+05	2
mean	0.217071	54356.993528	7.586507e+04	74.746530	1.659001e+05	2
std	0.412252	12971.314171	1.894478e+04	11.456636	9.422736e+05	2
min	0.000000	13320.000000	3.700000e+04	10.030000	-6.678296e+06	(
25%	0.000000	47145.000000	6.571700e+04	68.880000	0.000000e+00	(
50%	0.000000	53803.000000	7.094600e+04	76.800000	0.000000e+00	(
75%	0.000000	60413.000000	7.920175e+04	83.670000	3.500650e+04	6
max	1.000000	990572.000000	1.628992e+06	95.000000	9.652492e+07	1

In [65]:
```
final_data.describe(include = 'object').T
```

Out[65]:

	count	unique	top	freq
branch_id	233154	82	2	13138
supplier_id	233154	2953	18317	1432
manufacturer_id	233154	11	86	109534
state_id	233154	22	4	44870
employment_type	233154	3	Self employed	127635
mobileno_avl_flag	233154	1	1	233154
aadhar_flag	233154	2	1	195924
pan_flag	233154	2	0	215533
voterid_flag	233154	2	0	199360
driving_flag	233154	2	0	227735
passport_flag	233154	2	0	232658
new_accts_in_last_six_months	233154	26	0	181494
delinquent_accts_in_last_six_months	233154	14	0	214959
pri_no_of_accts	233154	108	0	116950
pri_active_accts	233154	40	0	137016
pri_overdue_accts	233154	22	0	206879
sec_no_of_accts	233154	37	0	227289
sec_active_accts	233154	23	0	229337
sec_overdue_accts	233154	9	0	231817
loan_tenure	233154	2	most_pref	230077
credit_hist_cat	233154	3	Low	214224
perform_cat	233154	7	0	129785
perform_score	233154	6	No Score	129785
preferred_age	233154	1	Negative	233154

In [66]:

```
train,test =split(final_data, test_size = 0.3, random_state = 12)
```

apply logistic regression

In [67]:

```
data_dummy = pd.get_dummies(final_data)
data_dummy.columns
```

Out[67]:
```
Index(['loan_default', 'disbursed_amount', 'asset_cost', 'ltv',
       'pri_current_balance', 'pri_sanctioned_amount', 'pri_disbursed_amount',
       'sec_current_balance', 'sec_sanctioned_amount', 'sec_disbursed_amount',
       ...
       'perform_cat_6', 'perform_cat_7', 'perform_cat_8', 'perform_score_High',
       'perform_score_Low', 'perform_score_Medium', 'perform_score_No Score',
       'perform_score_Very High', 'perform_score_Very Low',
       'preferred_age_Negative'],
      dtype='object', length=3392)
```

In [68]:
```
train, test = split(data_dummy, test_size = .30, random_state = 12)
train.shape

train.head(2)
X_train = train.drop('loan_default', axis = 1)
Y_train = train.loan_default
X_test = test.drop('loan_default', axis = 1)
Y_test = test.loan_default
lr = LogisticRegression()
lr.fit(X_train,Y_train)

pred = lr.predict(X_test)

print('Accuracy Score',accuracy_score(y_true = Y_test,y_pred = pred))
```

```
Accuracy Score 0.7854661386478333
```

In [69]:
```
print(classification_report(y_true=Y_test,y_pred = pred))
```

```
              precision    recall  f1-score   support

           0       0.79      1.00      0.88     54941
           1       0.50      0.00      0.00     15006

    accuracy                           0.79     69947
   macro avg       0.64      0.50      0.44     69947
weighted avg       0.72      0.79      0.69     69947
```