Capstone Project - Vehicle Loan Default Prediction

Pravat Kumar Sahu

Simplilearn - PGP DA FEB 2021 Cohort 1

The data is about Finance sector and related to vehicle loan taken by customers and loan account status. The data has 233154 observations and 41 variables. The dataframe is mix of interger, object, datetype. To begin with the study of the data, let's first import all necessary packages and import the dataset to the jupyter notebook.

In [1]:
```
import pandas as pd
import numpy as np
import os
import re
import datetime as dt
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from sklearn.model_selection import train_test_split as split
from sklearn.linear_model import LinearRegression
```

In [2]:
```
#Load the data using pandas.
importdata = pd.read_excel('/Users/priya/Pravat/Simplilearn Data Analytics/Class 5/project/project 2/data.xlsx')
importdata
```

Out[2]:

	UniqueID	disbursed_amount	asset_cost	ltv	branch_id	supplier_id	manufacturer_id	Current_pincode_ID	Date.of.Birth	Employmer
0	420825	50578	58400	89.55	67	22807	45	1441	1984-01-01	S
1	417566	53278	61360	89.63	67	22807	45	1497	1985-08-24	Self em
2	539055	52378	60300	88.39	67	22807	45	1495	1977-12-09	Self em
3	529269	46349	61500	76.42	67	22807	45	1502	1988-06-01	S
4	563215	43594	78256	57.50	67	22744	86	1499	1994-07-14	Self em
...	...	...	...	...	...	...	...	...	...	
233149	561031	57759	76350	77.28	5	22289	51	3326	1981-11-10	Self em
233150	649600	55009	71200	78.72	138	17408	51	3385	1992-10-15	Self em
233151	603445	58513	68000	88.24	135	23313	45	1797	1981-12-19	Self em

	UniqueID	disbursed_amount	asset_cost	ltv	branch_id	supplier_id	manufacturer_id	Current_pincode_ID	Date.of.Birth	Employmer
233152	442948	22824	40458	61.79	160	16212	48	96	1989-07-31	Self em
233153	545300	35299	72698	52.27	3	14573	45	17	1968-08-01	Self em

233154 rows × 41 columns

In [3]:
```
loandata = importdata
```

1. Preliminary Data analysis

In [4]:
```
#find the structure of the data
loandata.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
```

In [5]:
```
loandata.head(5)
```

Out[5]:

	UniqueID	disbursed_amount	asset_cost	ltv	branch_id	supplier_id	manufacturer_id	Current_pincode_ID	Date.of.Birth	Employment.Type
0	420825	50578	58400	89.55	67	22807	45	1441	1984-01-01	Salariec
1	417566	53278	61360	89.63	67	22807	45	1497	1985-08-24	Self employec
2	539055	52378	60300	88.39	67	22807	45	1495	1977-12-09	Self employec
3	529269	46349	61500	76.42	67	22807	45	1502	1988-06-01	Salariec
4	563215	43594	78256	57.50	67	22744	86	1499	1994-07-14	Self employec

5 rows × 41 columns

In [6]:
```
loandata.columns
```

Out[6]:
```
Index(['UniqueID', 'disbursed_amount', 'asset_cost', 'ltv', 'branch_id',
       'supplier_id', 'manufacturer_id', 'Current_pincode_ID', 'Date.of.Birth',
       'Employment.Type', 'DisbursalDate', 'State_ID', 'Employee_code_ID',
       'MobileNo_Avl_Flag', 'Aadhar_flag', 'PAN_flag', 'VoterID_flag',
       'Driving_flag', 'Passport_flag', 'PERFORM_CNS.SCORE',
       'PERFORM_CNS.SCORE.DESCRIPTION', 'PRI.NO.OF.ACCTS', 'PRI.ACTIVE.ACCTS',
       'PRI.OVERDUE.ACCTS', 'PRI.CURRENT.BALANCE', 'PRI.SANCTIONED.AMOUNT',
       'PRI.DISBURSED.AMOUNT', 'SEC.NO.OF.ACCTS', 'SEC.ACTIVE.ACCTS',
       'SEC.OVERDUE.ACCTS', 'SEC.CURRENT.BALANCE', 'SEC.SANCTIONED.AMOUNT',
```

```
        'SEC.DISBURSED.AMOUNT', 'PRIMARY.INSTAL.AMT', 'SEC.INSTAL.AMT',
        'NEW.ACCTS.IN.LAST.SIX.MONTHS', 'DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS',
        'AVERAGE.ACCT.AGE', 'CREDIT.HISTORY.LENGTH', 'NO.OF_INQUIRIES',
        'loan_default'],
       dtype='object')
```

In [7]:
```
#Check for null values in the data. Get the number of null values for each column.
loandata.isnull().sum()
```

Out[7]:
```
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 [8]:
```
#Missing values found in Employement type column only. As it is catagorial data, fill the missing values with Mode valu
loandata['Employment.Type'].fillna(loandata['Employment.Type'].mode()[0], inplace=True)
```

In [9]:
```
#verify after fill NA.
loandata.isnull().sum()
```

Out[9]:
```
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                       0
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
```

Alternatively the we can also drop missing records if number of missing data points are less and dont make any impact if ignored. To do this user can follow below code in pandas. loandata.dropna(axis = 0, inplace = True)

In [10]:
```
#Now let's check number of rows and coloumns in dataframe
loandata.shape
```

Out[10]:
```
(233154, 41)
```

In [11]:
```
#Let's check the if any duplicates in dataframe
loandata.duplicated().any()
```

Out[11]:
```
False
```

Result : As the result is False, we can conclude no duplicates in the dataframe

In [12]:
```
loandata.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                       233154 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
```

In [13]:

```
#Variable names in the data may not be in accordance with the identifier naming in Python, so let's change the variable
```

In [14]:

```
loandata = loandata.rename(columns={"Date.of.Birth":"Date_of_Birth",
                                    "Employment.Type":"Employment_Type",
                                    "PERFORM_CNS.SCORE":"PERFORM_CNS_SCORE",
                                    "PERFORM_CNS.SCORE.DESCRIPTION":"PERFORM_CNS_SCORE_DESCRIPTION",
                                    "PRI.NO.OF.ACCTS":"PRI_NO_OF_ACCTS",
                                    "PRI.ACTIVE.ACCTS":"PRI_ACTIVE_ACCTS",
                                    "PRI.OVERDUE.ACCTS":"PRI_OVERDUE_ACCTS",
                                    "PRI.CURRENT.BALANCE":"PRI_CURRENT_BALANCE",
                                    "PRI.SANCTIONED.AMOUNT":"PRI_SANCTIONED_AMOUNT",
                                    "PRI.DISBURSED.AMOUNT":"PRI_DISBURSED_AMOUNT",
                                    "SEC.NO.OF.ACCTS":"SEC_NO_OF_ACCTS",
                                    "SEC.ACTIVE.ACCTS":"SEC_ACTIVE_ACCTS",
                                    "SEC.OVERDUE.ACCTS":"SEC_OVERDUE_ACCTS",
                                    "SEC.CURRENT.BALANCE":"SEC_CURRENT_BALANCE",
                                    "SEC.SANCTIONED.AMOUNT":"SEC_SANCTIONED_AMOUNT",
                                    "SEC.DISBURSED.AMOUNT":"SEC_DISBURSED_AMOUNT",
                                    "PRIMARY.INSTAL.AMT":"PRIMARY_INSTAL_AMT",
                                    "SEC.INSTAL.AMT":"SEC_INSTAL_AMT",
```

```
                                     "NEW.ACCTS.IN.LAST.SIX.MONTHS":"NEW_ACCTS_IN_LAST_SIX_MONTHS",
                                     "DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS":"DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS",
                                     "AVERAGE.ACCT.AGE":"AVERAGE_ACCT_AGE",
                                     "CREDIT.HISTORY.LENGTH":"CREDIT_HISTORY_LENGTH",
                                     "NO.OF_INQUIRIES":"NO_OF_INQUIRIES"})
```

In [15]:

```
#Verify the Variable names after rename
loandata.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                       233154 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
```

2. Performing EDA

In [16]:
```
#Check the statistical description of the quantitative data variables
loandata.describe()
```

Out[16]:

	UniqueID	disbursed_amount	asset_cost	ltv	branch_id	supplier_id	manufacturer_id	Current_pincode_ID
count	233154.000000	233154.000000	2.331540e+05	233154.000000	233154.000000	233154.000000	233154.000000	233154.000000
mean	535917.573376	54356.993528	7.586507e+04	74.746530	72.936094	19638.635035	69.028054	3396.880247
std	68315.693711	12971.314171	1.894478e+04	11.456636	69.834995	3491.949566	22.141304	2238.147502
min	417428.000000	13320.000000	3.700000e+04	10.030000	1.000000	10524.000000	45.000000	1.000000
25%	476786.250000	47145.000000	6.571700e+04	68.880000	14.000000	16535.000000	48.000000	1511.000000
50%	535978.500000	53803.000000	7.094600e+04	76.800000	61.000000	20333.000000	86.000000	2970.000000
75%	595039.750000	60413.000000	7.920175e+04	83.670000	130.000000	23000.000000	86.000000	5677.000000
max	671084.000000	990572.000000	1.628992e+06	95.000000	261.000000	24803.000000	156.000000	7345.000000

8 rows × 35 columns

Above stats shows that the dataframe has 233154 unique customer id. Maximum loan disbursed amount is 990572, minimum is 13320 and averge loan disbursed amount is 54356. The standard devaition in loan disbursed amount is 12971.

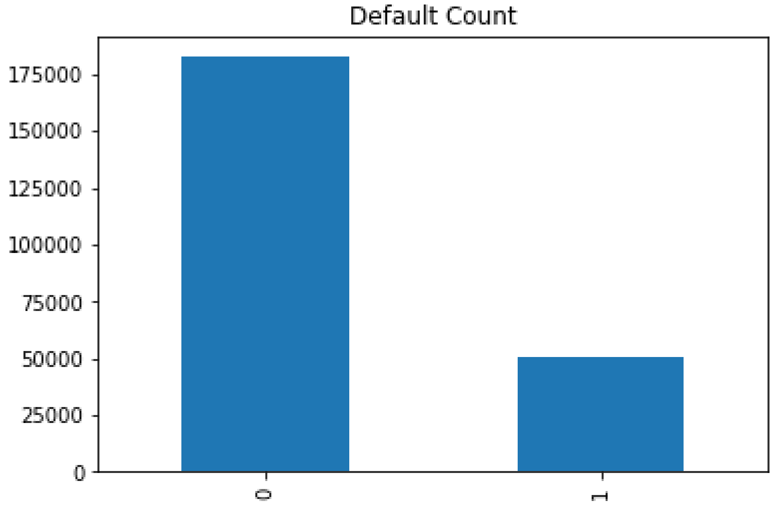
In [17]:
```
#How is the target variable distributed overall?
```

Target variables in the dataframe is loan default. The data type is a binary which has either 1 or 0 value. 1 indicates customer who are defaulted in loan repaymemt and 0 indicates who are not. so, to see how target variables are distributed we can use value_counts function as below.

In [18]:
```
print(loandata.loan_default.value_counts())
loandata.loan_default.value_counts().plot.bar()
plt.title('Default Count')
```

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

Out[18]:
```
Text(0.5, 1.0, 'Default Count')
```

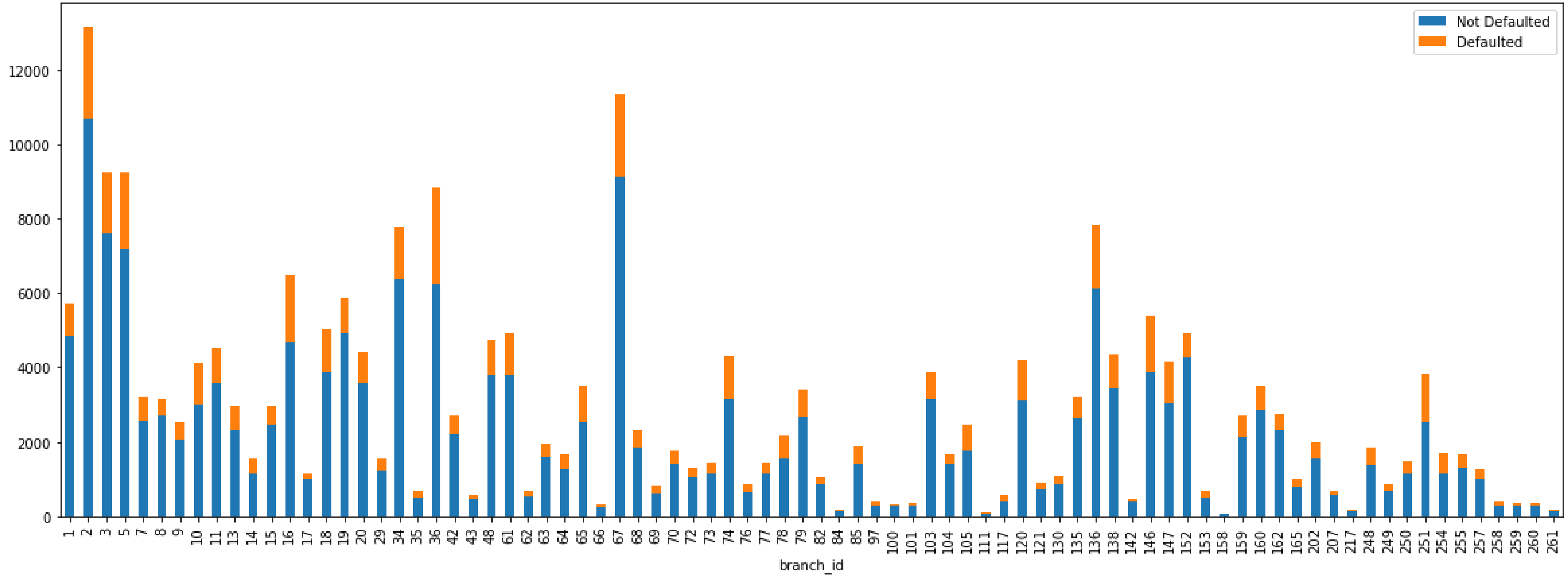


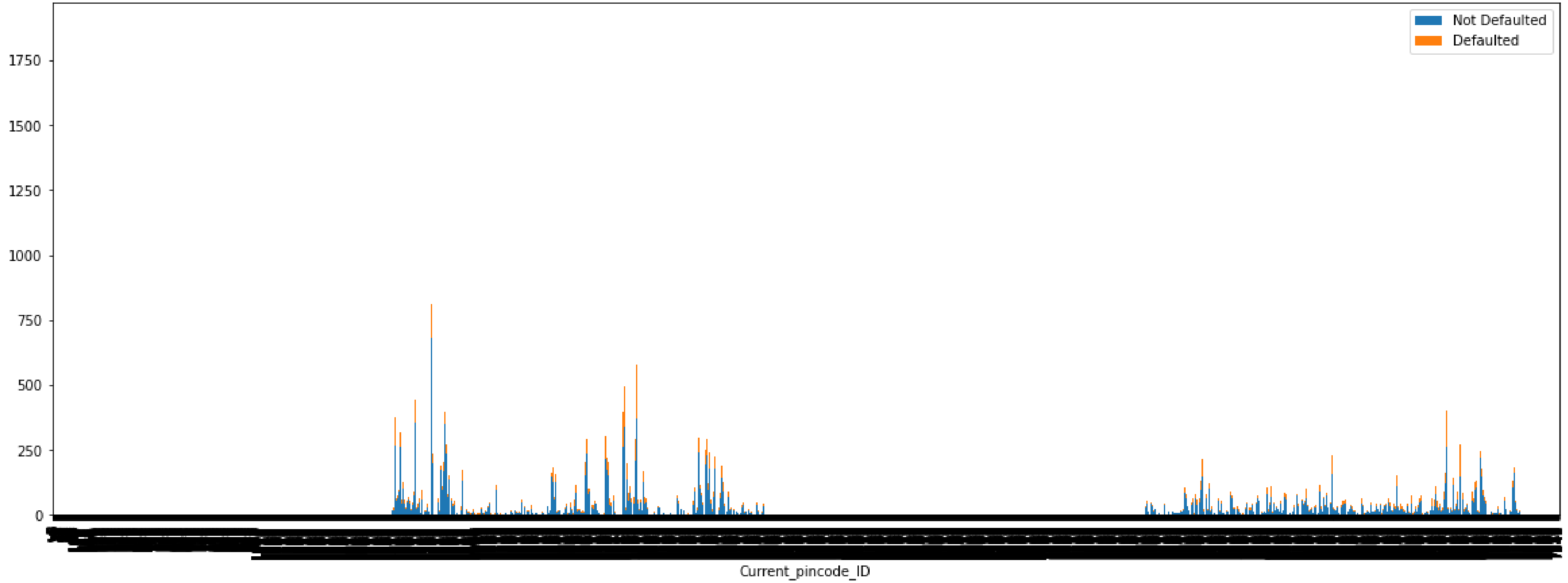
From the above result, we can say that majority that is roughly 78% customers are non defaulters while less portion that is 22% are defaulters.

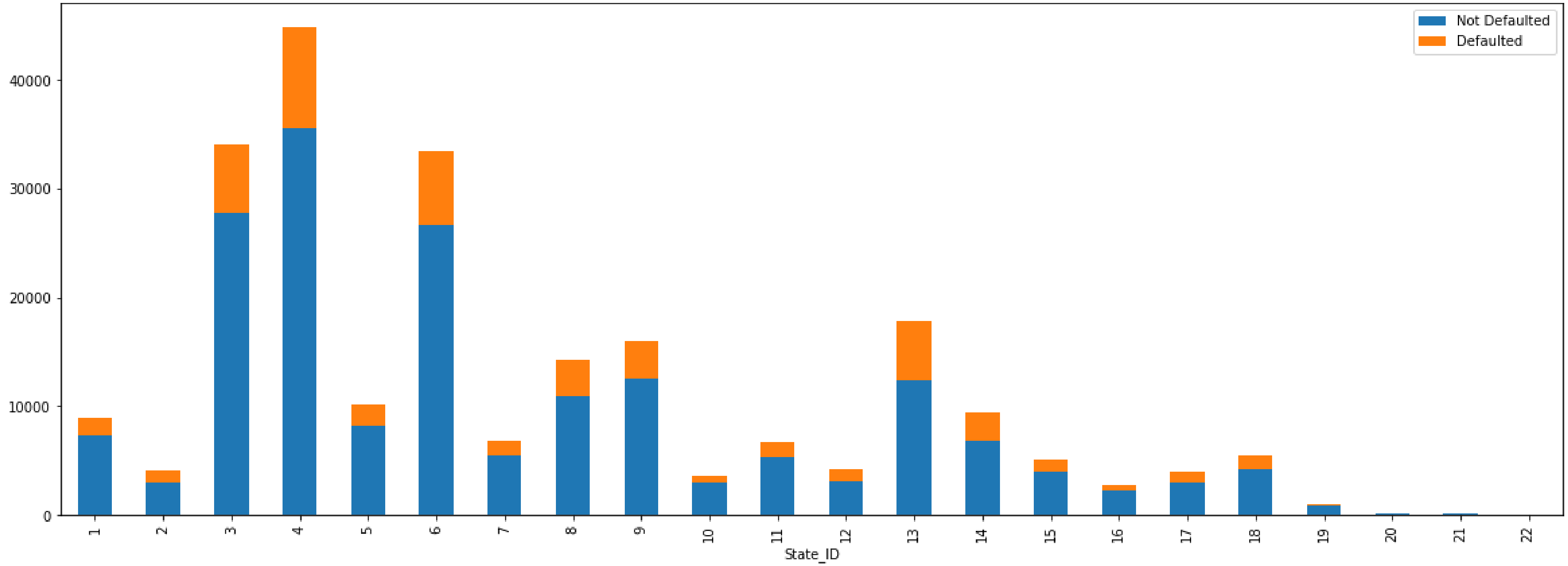
In [19]:
```
#Study the distribution of the target variable across the various categories such as branch, city, state, branch, suppl
```

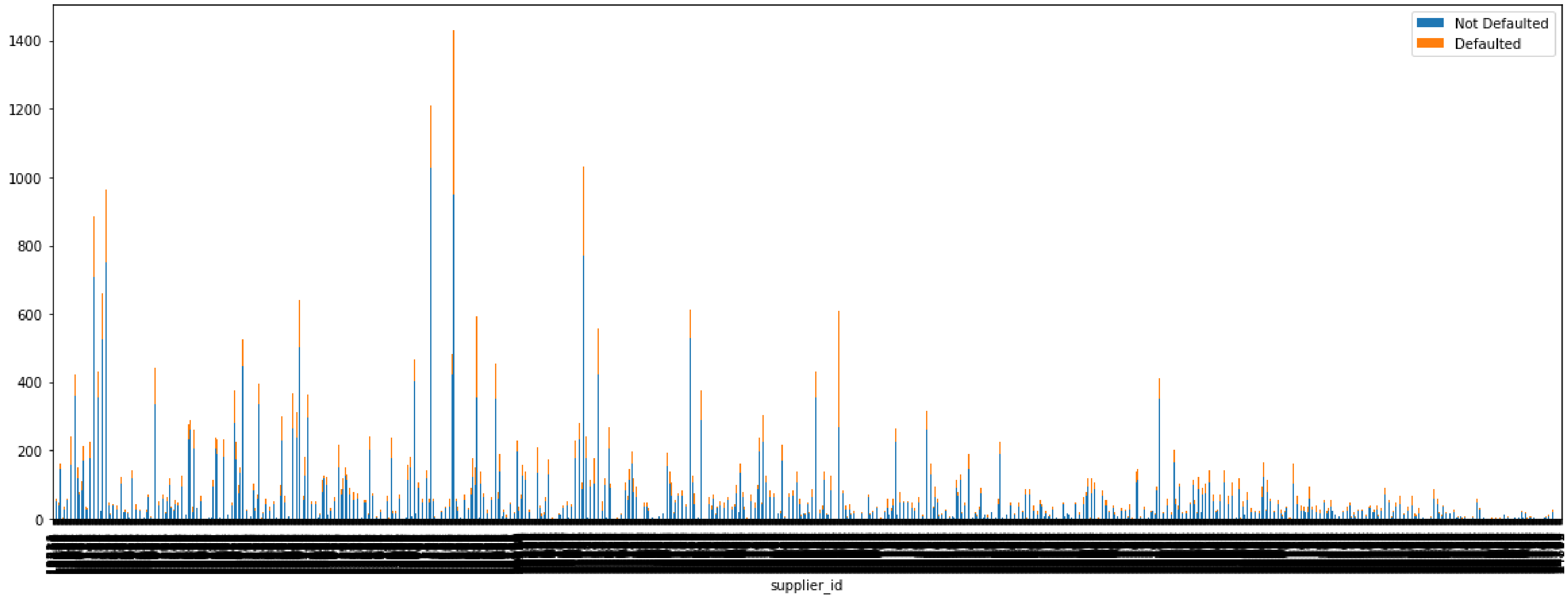
To study this, let's use for loop function and plot various stacked bar chart to check how defaulters and non defaulers related to various categorial variables are distribuited accross all branch, city, state, suppliers, manufaturers etc.

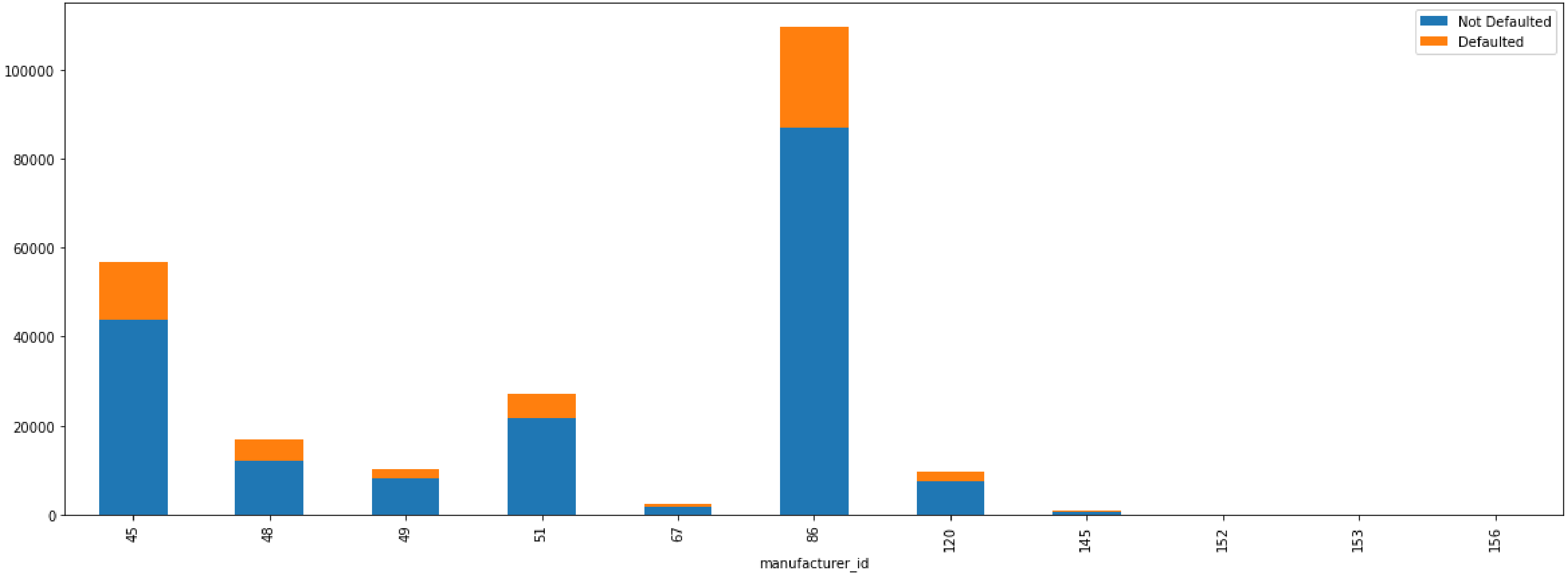
In [20]:
```
for i in ['branch_id','Current_pincode_ID','State_ID','supplier_id','manufacturer_id']:
    ct = pd.crosstab(loandata[i], loandata['loan_default'])
    ct.plot.bar(stacked = True,figsize=(20,7))
    plt.legend(labels=['Not Defaulted','Defaulted'])
    plt.show()
```


Not Defaulted
Defaulted
0
2000
4000
6000
8000
10000
12000
1 2 3 5 7 8 9 10 11 13 14 15 16 17 18 19 20 29 34 35 36 42 43 48 61 62 63 64 65 66 67 68 69 70 72 73 74 76 77 78 79 82 84 85 97 100 101 103 104 105 111 117 120 121 130 135 136 138 142 146 147 152 153 158 159 160 162 165 202 207 217 248 249 250 251 254 255 257 258 259 260 261
branch_id


Not Defaulted
Defaulted
0
250
500
750
1000
1250
1500
1750
Current_pincode_ID


Not Defaulted
Defaulted
40000
30000
20000
10000
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
State_ID

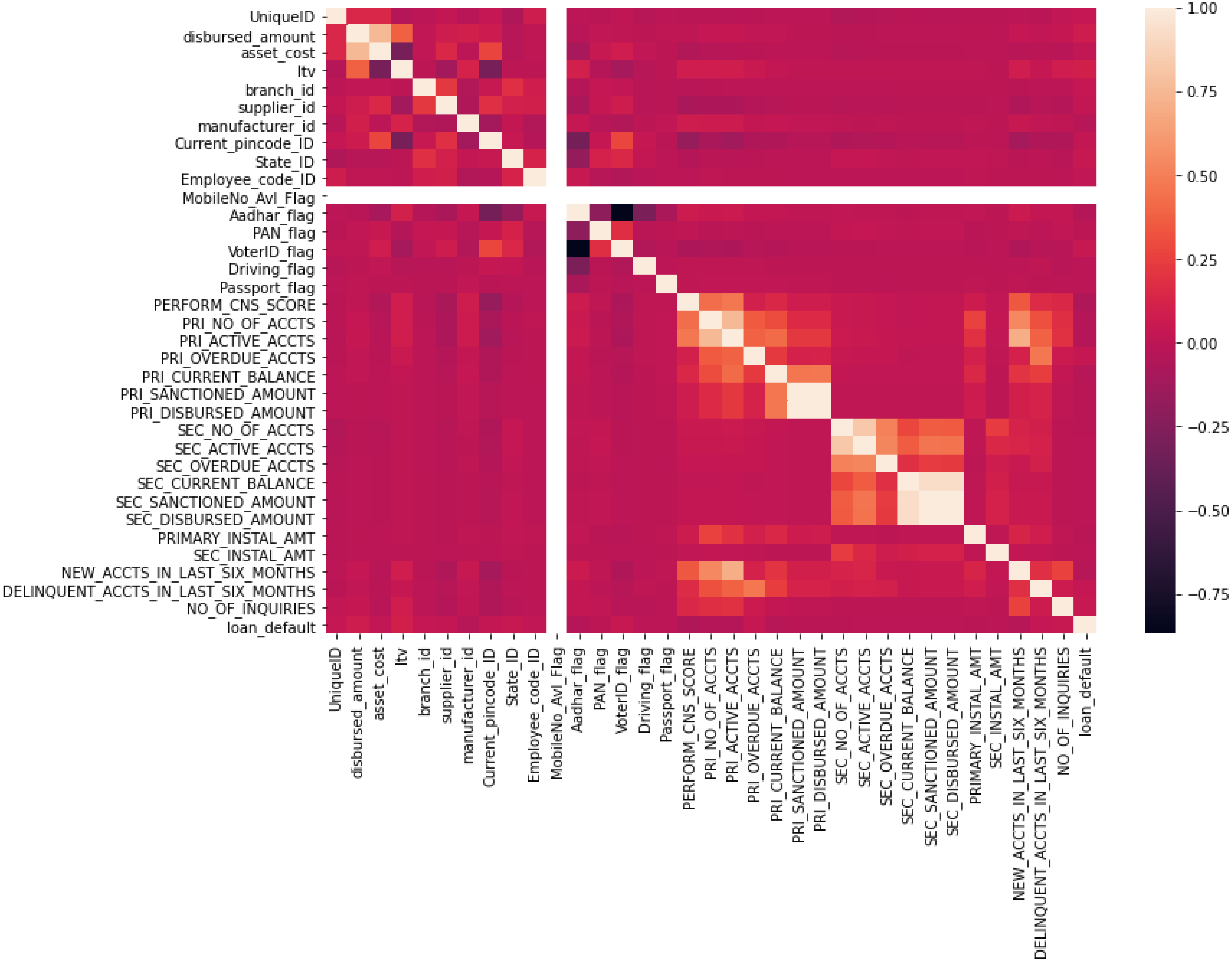

Not Defaulted
Defaulted
1400
1200
1000
800
600
400
200
0
supplier_id



From the stacked bar chart, Branch id 2, 36 and 67 are among the list of branch which have highest defaulters. State id 3, 4, 6 and 13 are among the list of states which have highest defaulters. Manufacturer id 86 have highest defaulters. For City and suppliers, it is difficult to infer how the target variables are distributed.We can draw a heat map to know the relationship between the target and independent variables.

In [21]:
```
plt.figure(figsize=(12,8))
sns.heatmap(loandata.corr())
```

Out[21]: <AxesSubplot:>



In [22]:
```
#What are the different employment types given in the data? Can a strategy be developed to fill in the missing values (
```

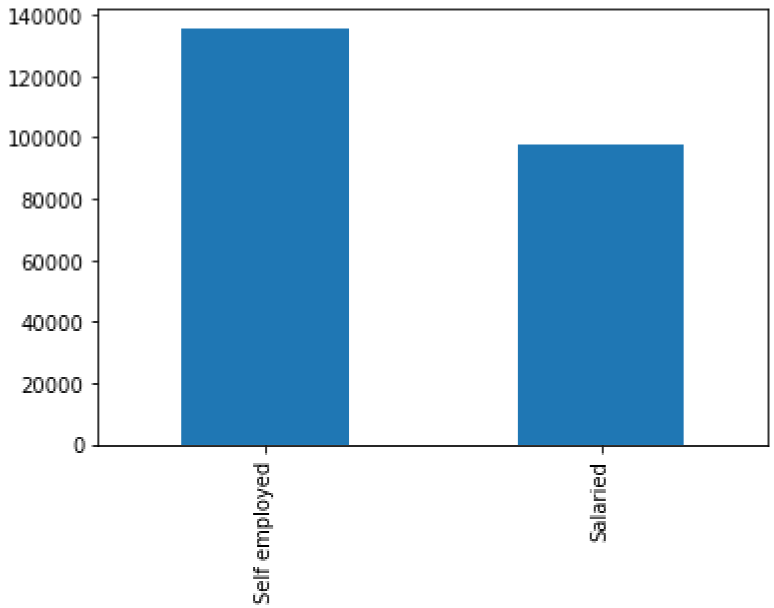
To get this let's use value count function on employment type and draw a bar chart to see how are the employment types in the dataframe.

In [23]:
```
loandata['Employment_Type'].value_counts()
```

Out[23]:
```
Self employed    135296
Salaried          97858
Name: Employment_Type, dtype: int64
```

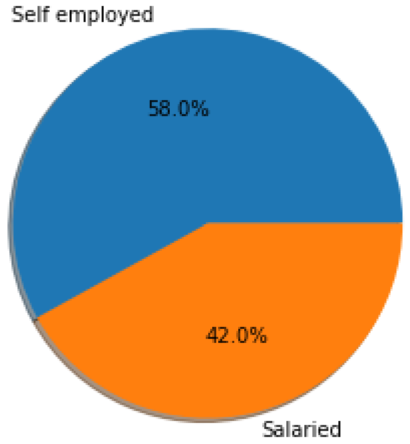
In [24]:
```
loandata['Employment_Type'].value_counts().plot(kind='bar')
```

Out[24]:
```
<AxesSubplot:>
```



In [25]:
```
# pie chart
labels = ['Self employed', 'Salaried']
sizes = loandata['Employment_Type'].value_counts()

fig1, ax1 = plt.subplots()
ax1.pie(sizes, labels=labels, autopct='%1.1f%%', shadow=True)
ax1.axis('equal')
plt.show()
```



Since we have already filled the mising value with Mode Value previously there is no missing value now so we can say that there are 2 levels or category in Employed Type variable, they are Self employment and Salaried.let's check the data and plot bar chart and pie chart to express how different types of employment defines defaulter and non-defaulters.

In [26]:
```
pct_loan_default = loandata['loan_default'].value_counts(normalize=True)*100
pct_loan_default
```

Out[26]:
```
0    78.292888
1    21.707112
Name: loan_default, dtype: float64
```

Total Defaulters are 21% and non defaulters are 78%. Let's first check the % defaulted customer for each employment category.

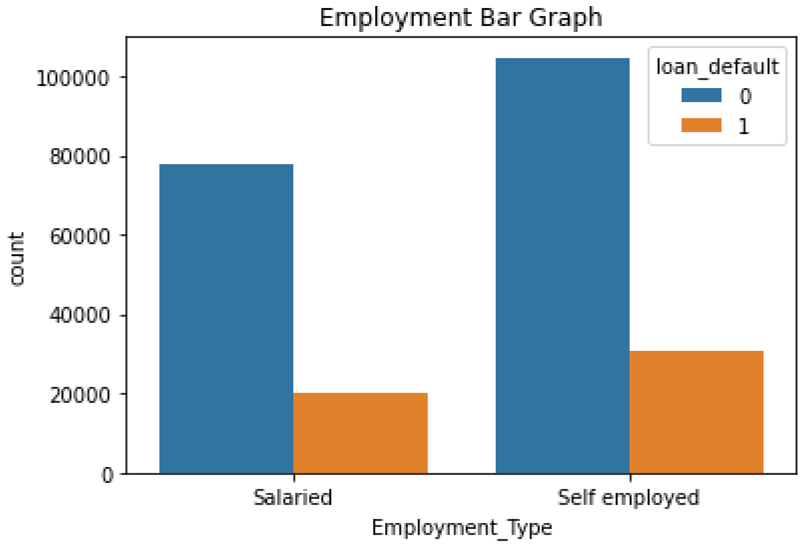
In [27]:
```
print('% of salaried customer only who have defaulted:',
     np.round(loandata[(loandata['Employment_Type']=='Salaried')
                                    & (loandata['loan_default']==1)].shape[0]/(loandata[loandata['Employment_Type']=='Salaried'].sha

print('% of self employed customer only who have defaulted:',
     np.round(loandata[(loandata['Employment_Type']=='Self employed')
                                    & (loandata['loan_default']==1)].shape[0]/(loandata[loandata['Employment_Type']=='Self employed'
```

```
% of salaried customer only who have defaulted: 20.35
% of self employed customer only who have defaulted: 22.69
```

In [28]:
```
# Let's Bar Chart to draw the employment vs loan default chart
sns.countplot(x='Employment_Type',hue='loan_default',data=loandata)
plt.title('Employment Bar Graph')
```

Out[28]:
```
Text(0.5, 1.0, 'Employment Bar Graph')
```



Now, check and express how both types of employment defines defaulter and non-defaulters, lets create a cross tab and plot pie chart.

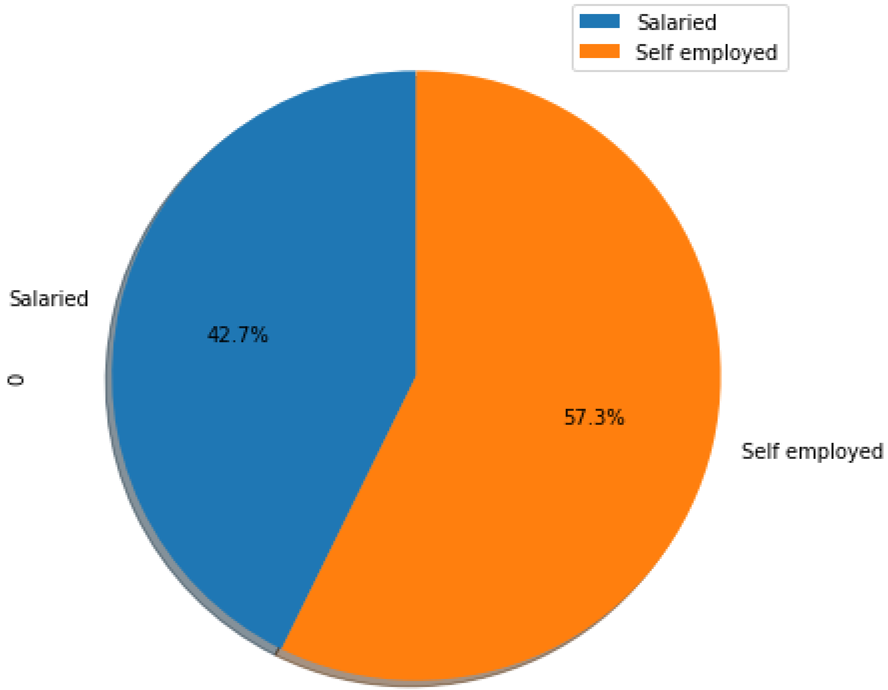
In [29]:
```
emp_loan=pd.crosstab(loandata['Employment_Type'],loandata['loan_default'])
emp_loan
```

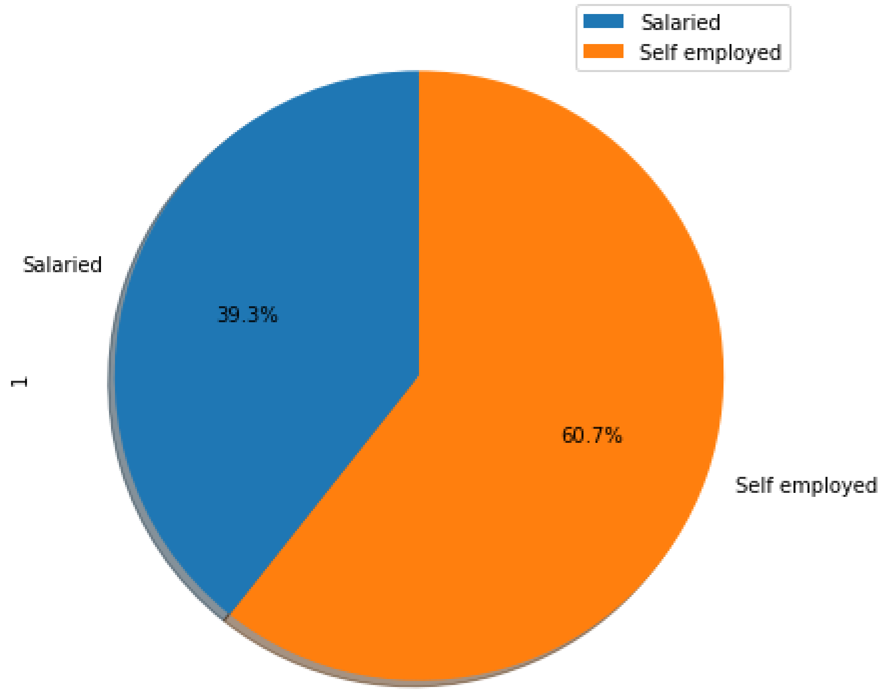
Out[29]:

loan_default	0	1
Employment_Type		
Salaried	77948	19910
Self employed	104595	30701

In [30]:
```
emp_loan.groupby(['Employment_Type']).sum().plot(kind='pie', subplots=True, shadow = True,startangle=90,
figsize=(15,10), autopct='%1.1f%%')
```

Out[30]:
```
array([<AxesSubplot:ylabel='0'>, <AxesSubplot:ylabel='1'>], dtype=object)
```





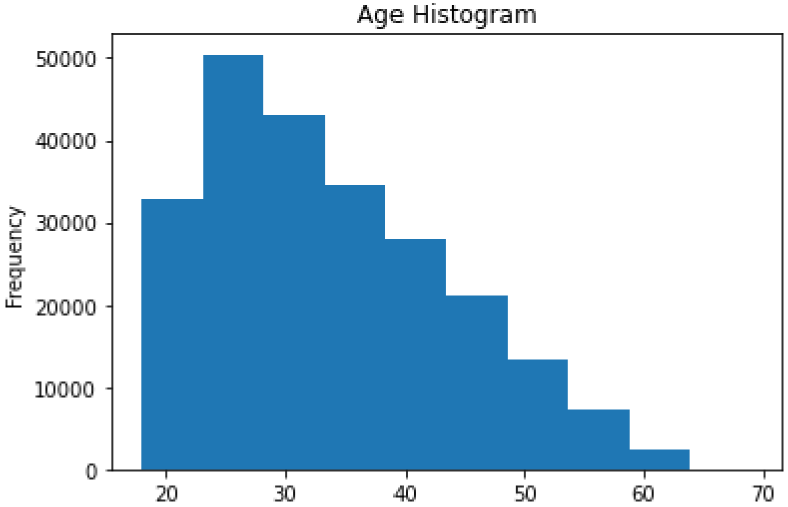
Above pie chart shows that arround 57.3% of the customers self employed and 42.7% of the customers salaried by profession are non-defaulters while 60.7% of the customers who are self employed and 39.3% of the customers salaries by profession are defaulters.

In [31]:
```
#Has age got something to do with defaulting? What is the distribution of age w.r.t. to defaulters and non-defaulters?
```

We have Date of Birth of the customer and the date of disbursal of loan from which we can to calculate the age of the customer at the the time of disbursement of loan. Let's draw a histogram to observe the distribution of the ages in the dataframe.

In [32]:
```
loandata['age'] = pd.DatetimeIndex(loandata['DisbursalDate']).year - pd.DatetimeIndex(loandata['Date_of_Birth']).year
loandata['age'].plot.hist()
plt.title('Age Histogram')
```

Out[32]:
```
Text(0.5, 1.0, 'Age Histogram')
```



Histogram plot - Noticed that customers age are between 18-69.

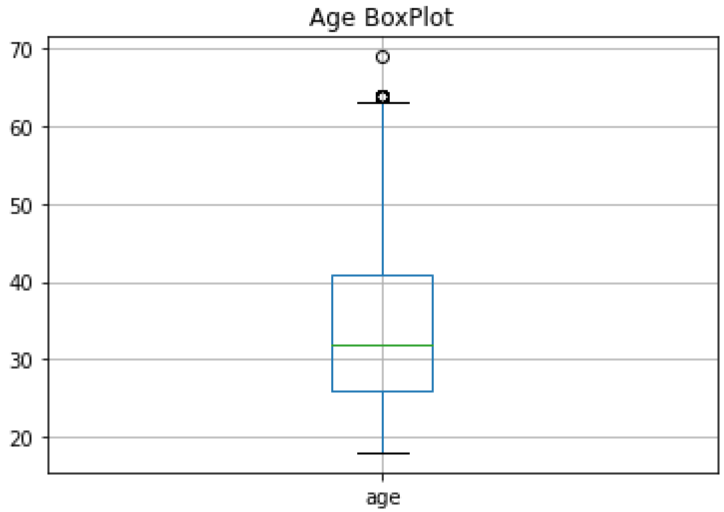
In [33]:
```
loandata.age.describe()
```

Out[33]:
```
count    233154.000000
mean         34.100946
std           9.805992
min          18.000000
25%          26.000000
50%          32.000000
75%          41.000000
max          69.000000
Name: age, dtype: float64
```

We can plot a box plot to see the distribution of the age groups as well.

In [34]:
```
loandata.boxplot('age')
plt.title('Age BoxPlot')
```

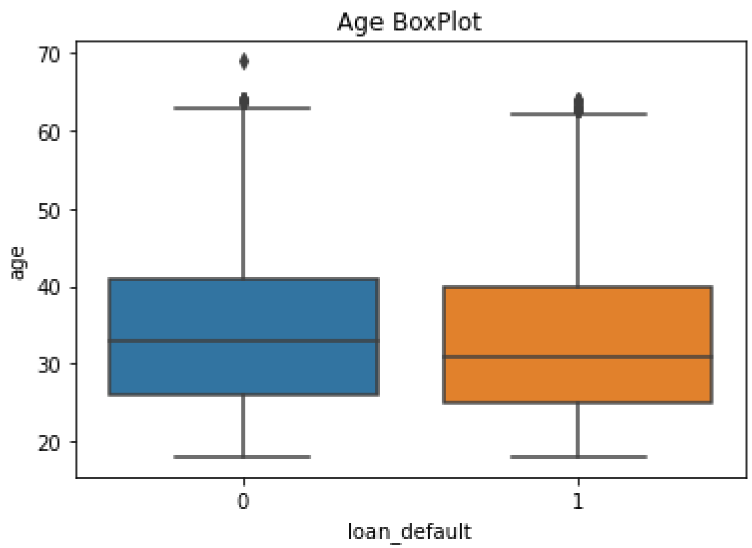
Out[34]:
```
Text(0.5, 1.0, 'Age BoxPlot')
```



From the box plot, it is obvervserd that avergae age is 34 and lowest age of customer taking loan is 18 while highest is 69. Age above 60 are outliers.Now, lets draw the relation between between age and loan default variable.

In [35]:
```
sns.boxplot(x='loan_default', y='age',data=loandata)
plt.title('Age BoxPlot')
```

Out[35]: Text(0.5, 1.0, 'Age BoxPlot')



From the box plot, observed that age are almost same for defaulters and non defaulters.

```
In [36]: #What type of ID was presented by most of the customers as proof?
```

To find out this which document was presented most, we can do value_counts for each document type variables.

```
In [37]: print(loandata['Aadhar_flag'].value_counts())
         print(loandata['PAN_flag'].value_counts())
         print(loandata['VoterID_flag'].value_counts())
         print(loandata['Driving_flag'].value_counts())
         print(loandata['Passport_flag'].value_counts())
```

```
1    195924
0     37230
Name: Aadhar_flag, dtype: int64
0    215533
1     17621
Name: PAN_flag, dtype: int64
0    199360
1     33794
Name: VoterID_flag, dtype: int64
0    227735
1      5419
Name: Driving_flag, dtype: int64
0    232658
1       496
Name: Passport_flag, dtype: int64
```

```
In [38]: print(loandata['Aadhar_flag'].value_counts(normalize=True)*100)
         print(loandata['PAN_flag'].value_counts(normalize=True)*100)
         print(loandata['VoterID_flag'].value_counts(normalize=True)*100)
         print(loandata['Driving_flag'].value_counts(normalize=True)*100)
         print(loandata['Passport_flag'].value_counts(normalize=True)*100)
```

```
1    84.032013
0    15.967987
Name: Aadhar_flag, dtype: float64
0    92.442334
1     7.557666
Name: PAN_flag, dtype: float64
0    85.505717
1    14.494283
Name: VoterID_flag, dtype: float64
0    97.675785
1     2.324215
Name: Driving_flag, dtype: float64
0    99.787265
1     0.212735
Name: Passport_flag, dtype: float64
```

From the above result, we can infer that Aadhar card is the document which was presented most by customer (195924 customers) followed by Voter id (33794 customer)

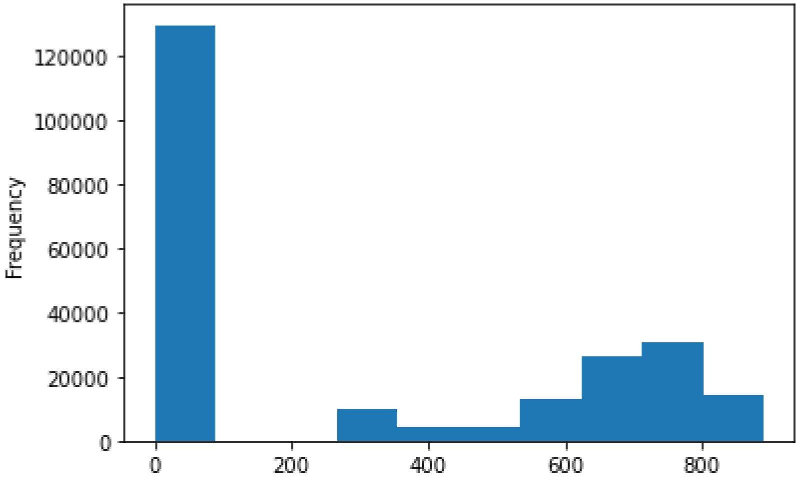
In [39]:
```
#Study the credit bureau score distribution. How is the distribution for defaulters vs non-defaulters? Explore in detai
```

PERFORM_CNS_SCORE is the variable which indicates Bureau or CIBIL Score for each customer. Let's study this by checking basic stat function.

In [40]:
```
loandata['PERFORM_CNS_SCORE'].describe()
```

Out[40]:
```
count    233154.000000
mean        289.462994
std         338.374779
min           0.000000
25%           0.000000
50%           0.000000
75%         678.000000
max         890.000000
Name: PERFORM_CNS_SCORE, dtype: float64
```

In [41]:
```
loandata['PERFORM_CNS_SCORE'].plot(kind='hist')
plt.show()
```



We can see that in the distribution, the avergae score is 289.46 and maximum score is 890. The minimum score is 0 which indicates there is no score available for that customer.Now let's filter defaulter and non defaulters and separetely study their score.

In [42]:
```
cibil_non_default = loandata[loandata['loan_default']==0]['PERFORM_CNS_SCORE']
cibil_default = loandata[loandata['loan_default']==1]['PERFORM_CNS_SCORE']
```

In [43]:
```
pd.DataFrame([cibil_non_default.describe(), cibil_default.describe()], index=['non_defaulters','defaulters'])
```

Out[43]:

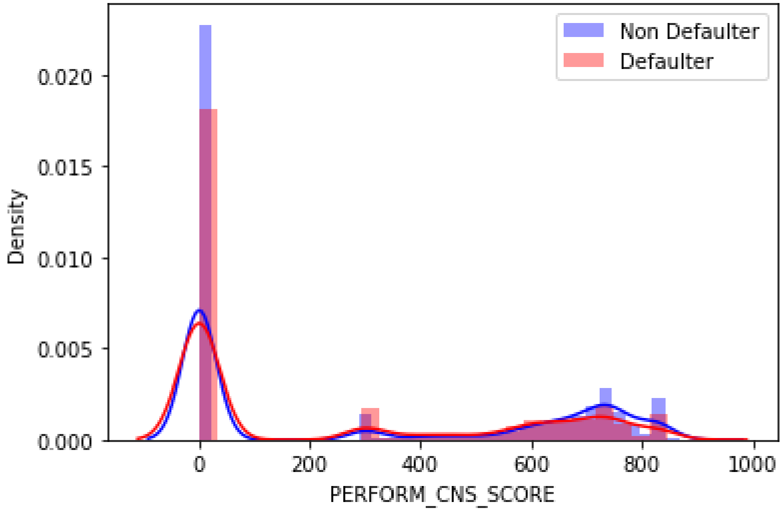
	count	mean	std	min	25%	50%	75%	max
non_defaulters	182543.0	299.784270	342.883794	0.0	0.0	15.0	690.0	890.0
defaulters	50611.0	252.236372	318.826242	0.0	0.0	0.0	610.0	879.0

We can observe a difference in the mean and median cibil scores among the defaulters and non defaulters. The mean and median in cibil scores are higher for non defaulters.

In [44]:
```
sns.distplot( a = cibil_non_default, color='blue', label = 'Non Defaulter')
sns.distplot(a = cibil_default, color='red', label = 'Defaulter')

plt.legend()
plt.show()
```

```
/Users/priya/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/Users/priya/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

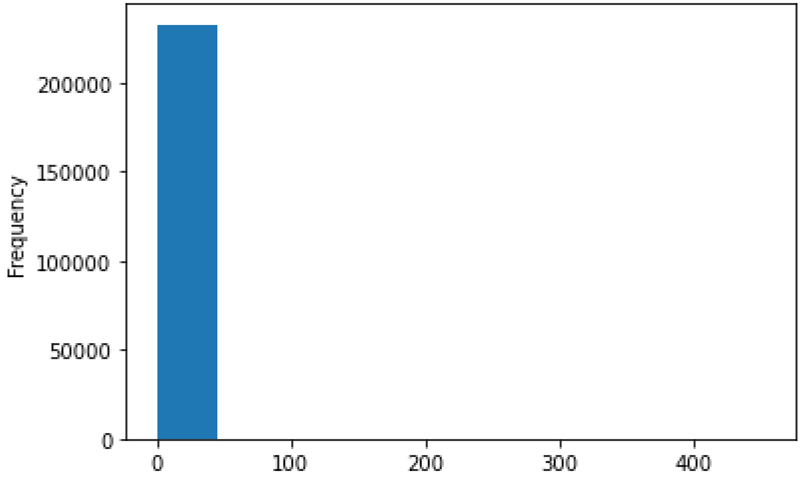


Above displot indicates the CIBIL score distribution is looking almost similar for defaulters and non defaulters customers.

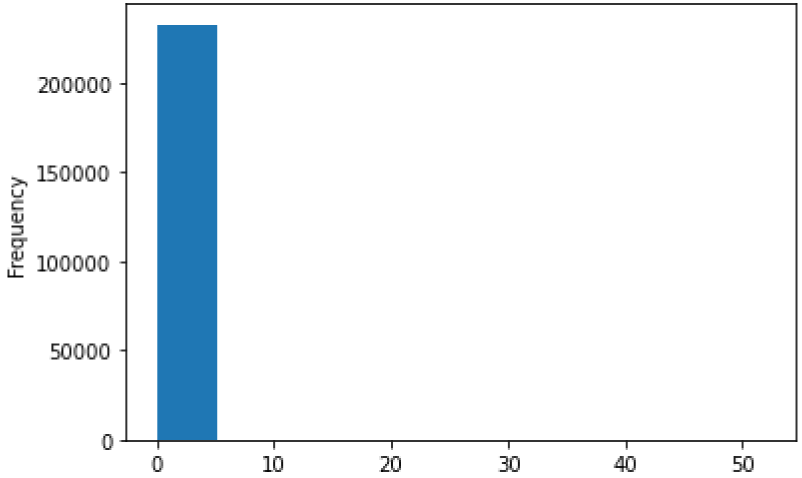
In [45]:
```
#Explore the primary and secondary account details. Is the information in some way related to loan default probability
```

To study this, let's check the histo plot for primary and secondary account.

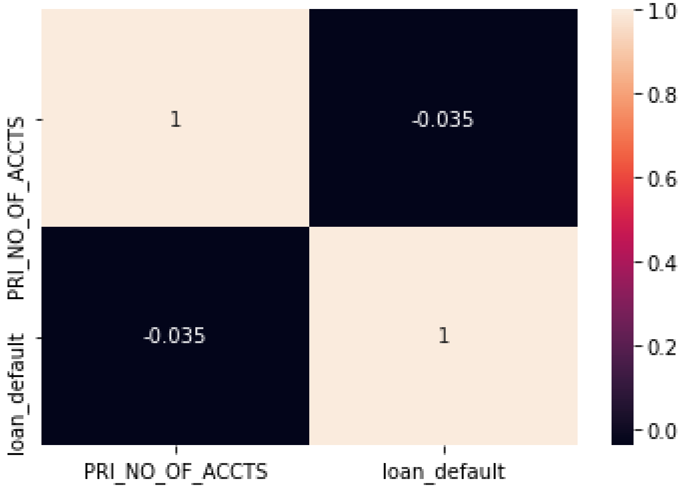
In [46]:
```
loandata['PRI_NO_OF_ACCTS'].plot(kind='hist')
plt.show()
```



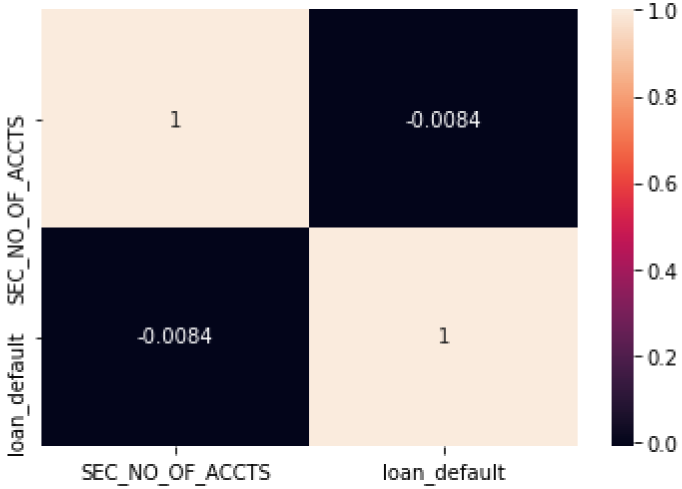
In [47]:
```
loandata['SEC_NO_OF_ACCTS'].plot(kind='hist')
plt.show()
```



In [48]:
```
#Checking the correlation between primary and loan deafult vairable
sns.heatmap(loandata[['PRI_NO_OF_ACCTS','loan_default']].corr(),annot=True)
plt.show()
```



In [49]:
```
#Checking the correlation between seondary and loan deafult vairable
sns.heatmap(loandata[['SEC_NO_OF_ACCTS','loan_default']].corr(),annot=True)
plt.show()
```



There is no correlation between no of primary or no of secondary account with loan default variable.Now, let's find out if this information is some way related to loan default probability.

In [50]:
```
pri_acct_non_default = loandata[loandata['loan_default']==0]['PRI_NO_OF_ACCTS']
pri_acct_default = loandata[loandata['loan_default']==1]['PRI_NO_OF_ACCTS']
```

In [51]:
```
pd.DataFrame([pri_acct_non_default.describe(), pri_acct_default.describe()], index=['non_defaulters','defaulters'])
```

Out[51]:

	count	mean	std	min	25%	50%	75%	max
non_defaulters	182543.0	2.538038	5.261142	0.0	0.0	1.0	3.0	354.0
defaulters	50611.0	2.089328	5.040134	0.0	0.0	0.0	2.0	453.0

In [52]:

```
sec_acct_non_default = loandata[loandata['loan_default']==0]['SEC_NO_OF_ACCTS']
sec_acct_default = loandata[loandata['loan_default']==1]['SEC_NO_OF_ACCTS']
```

In [53]:

```
pd.DataFrame([sec_acct_non_default.describe(), sec_acct_default.describe()], index=['non_defaulters','defaulters'])
```

Out[53]:

	count	mean	std	min	25%	50%	75%	max
non_defaulters	182543.0	0.061848	0.651657	0.0	0.0	0.0	0.0	52.0
defaulters	50611.0	0.049100	0.527358	0.0	0.0	0.0	0.0	38.0

Observed that for customers having primary accounts are maximum defaulters and customers having secondary accounts are less defaulters.

In [54]:

```
#Is there a difference between the sanctioned and disbursed amount of primary & secondary loans. Study the difference b
```

We value count function and find oout the % of the data for the sanctioned and disbursed amount for both primary and secondary account and see if there are any differnces.

In [55]:

```
pri_sanc_amt_counts = loandata['PRI_SANCTIONED_AMOUNT'].value_counts()
pri_sanc_amt_counts_percent = loandata['PRI_SANCTIONED_AMOUNT'].value_counts(normalize=True)*100

pd.DataFrame({'counts':pri_sanc_amt_counts,'percent_of_data':pri_sanc_amt_counts_percent})
```

Out[55]:

	counts	percent_of_data
0	138096	59.229522
50000	1503	0.644638
30000	1450	0.621907
100000	974	0.417750
25000	946	0.405740
...	...	...
114802	1	0.000429
1122414	1	0.000429

	counts	percent_of_data
1372970	1	0.000429
57430	1	0.000429
1134550	1	0.000429

44390 rows × 2 columns

In [56]:
```
pri_disb_amt_counts = loandata['PRI_DISBURSED_AMOUNT'].value_counts()
pri_disb_amt_counts_percent = loandata['PRI_DISBURSED_AMOUNT'].value_counts(normalize=True)*100

pd.DataFrame({'counts':pri_disb_amt_counts,'percent_of_data':pri_disb_amt_counts_percent})
```

Out[56]:

	counts	percent_of_data
0	138204	59.275843
50000	1398	0.599604
30000	1344	0.576443
100000	949	0.407027
40000	794	0.340547
...	...	...
417025	1	0.000429
898300	1	0.000429
1641300	1	0.000429
341228	1	0.000429
81432	1	0.000429

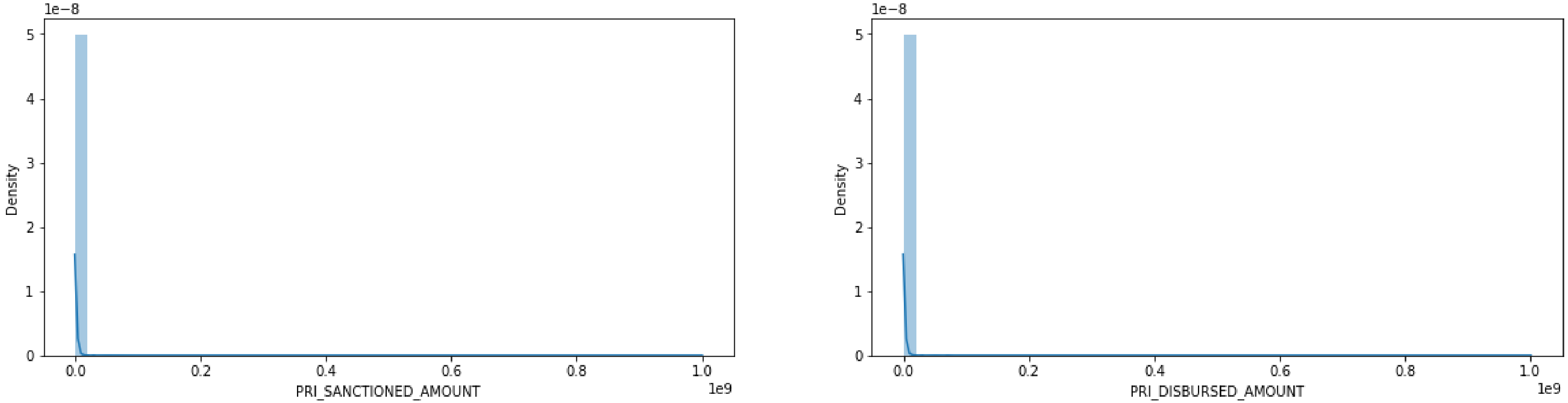
47909 rows × 2 columns

In [57]:
```
pri_acct_loan_amt =['PRI_SANCTIONED_AMOUNT', 'PRI_DISBURSED_AMOUNT']
```

In [58]:
```
count = 1
plt.figure(figsize=(20,10))
```

```
for i in pri_acct_loan_amt:
    plt.subplot(2,2,count)
    sns.distplot(loandata[i])
    count += 1
plt.show()
```

```
/Users/priya/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/Users/priya/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



For primary account holder customers there is not much differnce between sanctioned amount and disbursal amount as arround 59% of the accounts no amount was sanctioned or disbursed.

In [59]:
```
sec_sanc_amt_counts = loandata['SEC_SANCTIONED_AMOUNT'].value_counts()
sec_sanc_amt_counts_percent = loandata['SEC_SANCTIONED_AMOUNT'].value_counts(normalize=True)*100

pd.DataFrame({'counts':sec_sanc_amt_counts,'percent_of_data':sec_sanc_amt_counts_percent})
```

Out[59]:

	counts	percent_of_data
0	229418	98.397626
50000	83	0.035599
100000	61	0.026163

	counts	percent_of_data
30000	44	0.018872
40000	39	0.016727
...	...	...
232096	1	0.000429
51912	1	0.000429
1682532	1	0.000429
224700	1	0.000429
2349509	1	0.000429

2223 rows × 2 columns

In [60]:
```
sec_disb_amt_counts = loandata['SEC_DISBURSED_AMOUNT'].value_counts()
sec_disb_amt_counts_percent = loandata['SEC_DISBURSED_AMOUNT'].value_counts(normalize=True)*100

pd.DataFrame({'counts':sec_disb_amt_counts,'percent_of_data':sec_disb_amt_counts_percent})
```

Out[60]:

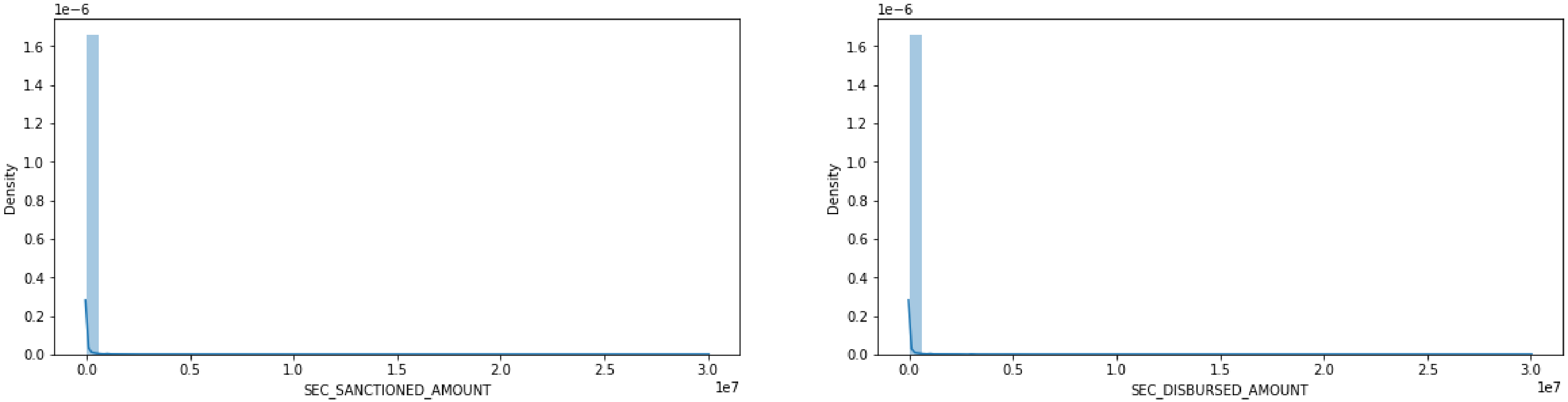
	counts	percent_of_data
0	229450	98.411350
50000	59	0.025305
100000	47	0.020158
200000	36	0.015440
40000	31	0.013296
...	...	...
654829	1	0.000429
19931	1	0.000429
48581	1	0.000429
91390	1	0.000429
74596	1	0.000429

2553 rows × 2 columns

In [61]:
```
sec_acct_loan_amt =['SEC_SANCTIONED_AMOUNT', 'SEC_DISBURSED_AMOUNT']
```

In [62]:
```
count=1
plt.figure(figsize=(20,10))
for i in sec_acct_loan_amt:
    plt.subplot(2,2,count)
    sns.distplot(loandata[i])
    count+=1
plt.show()
```

```
/Users/priya/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/Users/priya/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



For secondary account holder customers too there is not much differnce between sanctioned amount and disbursal amount as arround 98% of the accounts no amount was sanctioned or disbursed.Hypothesis Testing : Is there a any differnce between sanctioned and disbursed amount for Primary account and secondary account? Alternate Hypthoesis (Ha) : mu(sanctioned) - mu(disbursed) = 0, (there is no differnce between sanctioned and disbursed amount for Primary and secondary account) Null Hypothesis (H0) : mu(sanctioned) - mu(disbursed) <= 0, (there is differnce between sanctioned and disbursed amount for Primary and secondary account)

In [63]:
```
stats.ttest_ind(loandata.PRI_SANCTIONED_AMOUNT, loandata.PRI_DISBURSED_AMOUNT)
```

Out[63]:
```
Ttest_indResult(statistic=0.06292770557575765, pvalue=0.9498240996726557)
```

In [64]:
```
stats.ttest_ind(loandata.SEC_SANCTIONED_AMOUNT, loandata.SEC_DISBURSED_AMOUNT)
```

Out[64]:
```
Ttest_indResult(statistic=0.21643737250785414, pvalue=0.8286469322026462)
```

From the above T test result, it is oberved that for both primary and secondary account case the P value is greater than alpha 0.05, so we can conclude that we failed to reject null hypothesis that means there are differnces in sanctioned and disbursed amount for both primary and secondary account.

In [65]:
```
#Do customer who make higher no. of enquiries end up being higher risk candidates?
```

first let's identify how many customers or % of customers have made an inquiry before taking a loan.

In [66]:
```
enquiries_counts = loandata['NO_OF_INQUIRIES'].value_counts()
enquiries_counts_percent = loandata['NO_OF_INQUIRIES'].value_counts(normalize=True)*100

pd.DataFrame({'counts':enquiries_counts,'percent_of_data':enquiries_counts_percent})
```

Out[66]:

	counts	percent_of_data
0	201961	86.621289
1	22285	9.558060
2	5409	2.319926
3	1767	0.757868
4	760	0.325965
5	343	0.147113
6	239	0.102507
7	135	0.057902
8	105	0.045035
9	44	0.018872
10	34	0.014583
11	15	0.006434
12	14	0.006005
14	8	0.003431
15	7	0.003002

	counts	percent_of_data
13	6	0.002573
19	6	0.002573
17	4	0.001716
18	4	0.001716
16	3	0.001287
28	1	0.000429
20	1	0.000429
22	1	0.000429
23	1	0.000429
36	1	0.000429

Above statistics shows that most (approx 87%) of the customers have not made any enquiries regarding loans.

In [67]:

```
no_of_loan_inquiries = pd.crosstab(index=loandata['NO_OF_INQUIRIES'], columns=loandata['loan_default'])
no_of_loan_inquiries['pct_default'] = (no_of_loan_inquiries[1]/no_of_loan_inquiries.sum(axis=1))*100
no_of_loan_inquiries
```

Out[67]:

loan_default	0	1	pct_default
NO_OF_INQUIRIES			
0	159404	42557	21.071890
1	16844	5441	24.415526
2	3918	1491	27.565169
3	1250	517	29.258630
4	526	234	30.789474
5	212	131	38.192420
6	148	91	38.075314
7	80	55	40.740741
8	61	44	41.904762

loan_default	0	1	pct_default
NO_OF_INQUIRIES			
9	30	14	31.818182
10	23	11	32.352941
11	8	7	46.666667
12	10	4	28.571429
13	2	4	66.666667
14	6	2	25.000000
15	3	4	57.142857
16	3	0	0.000000
17	4	0	0.000000
18	2	2	50.000000
19	4	2	33.333333
20	1	0	0.000000
22	1	0	0.000000
23	1	0	0.000000
28	1	0	0.000000
36	1	0	0.000000

From the above result, we can infer that except for majority cases, as the number of enquires increase, there is an increase in the default percentage of customers and so being end up being higher risk candidates for the bank.

In [68]:
```
#Is credit history, i.e. new loans in last six months, loans defaulted in last six months, time since first loan, etc.,
```

Before we start our exploration, let's first create a function to replace alpha numeric values in CREDIT_HISTORY_LENGTH vairaible and change them to months. This will also change the data type to float type from object type.

In [69]:
```
def duration(dur):
    yrs = int(dur.split(' ')[0].replace('yrs',''))
    mon = int(dur.split(' ')[1].replace('mon',''))
    return yrs*12+mon
```

In [70]:
```
loandata['CREDIT_HISTORY_LENGTH'] = loandata['CREDIT_HISTORY_LENGTH'].apply(duration)
```

In [71]:
```
#verify to check the data type after function apply
loandata['CREDIT_HISTORY_LENGTH'].describe()
```

Out[71]:
```
count    233154.000000
mean         16.252404
std          28.581255
min           0.000000
25%           0.000000
50%           0.000000
75%          24.000000
max         468.000000
Name: CREDIT_HISTORY_LENGTH, dtype: float64
```

Now, let's see how the target variable is related to CREDIT_HISTORY_LENGTH variable.

In [72]:
```
credit_non_default = loandata[loandata['loan_default'] == 0]['CREDIT_HISTORY_LENGTH']
credit_default = loandata[loandata['loan_default'] == 1]['CREDIT_HISTORY_LENGTH']
```

In [73]:
```
pd.DataFrame([credit_non_default.describe(), credit_default.describe()], index=['non_defaulters','defaulters'])
```

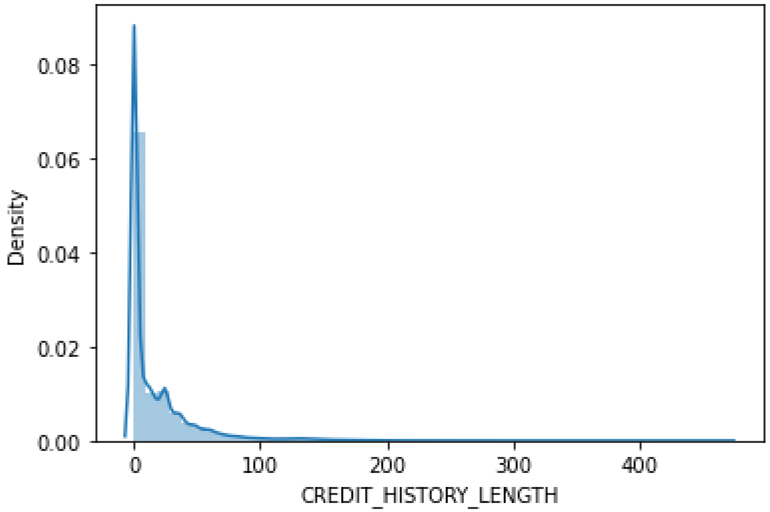
Out[73]:

	count	mean	std	min	25%	50%	75%	max
non_defaulters	182543.0	16.886377	29.342245	0.0	0.0	0.0	24.0	449.0
defaulters	50611.0	13.965798	25.519395	0.0	0.0	0.0	21.0	468.0

Above stats shows that the mean and standard deviation are higher for non defaulters customers.

In [74]:
```
sns.distplot(loandata['CREDIT_HISTORY_LENGTH'])
plt.show()
```

```
/Users/priya/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



From the displot observed that CREDIT_HISTORY_LENGTH is Highly right skewed.

In [75]:
```
new_acct_counts = loandata['NEW_ACCTS_IN_LAST_SIX_MONTHS'].value_counts()
new_acct_counts_percent = loandata['NEW_ACCTS_IN_LAST_SIX_MONTHS'].value_counts(normalize=True)*100

pd.DataFrame({'counts':new_acct_counts,'percent_of_data':new_acct_counts_percent})
```

Out[75]:

	counts	percent_of_data
0	181494	77.842971
1	32099	13.767295
2	11015	4.724345
3	4458	1.912041
4	1957	0.839359
5	964	0.413461
6	480	0.205873
7	302	0.129528
8	147	0.063048
9	79	0.033883
10	55	0.023590

	counts	percent_of_data
11	31	0.013296
12	20	0.008578
13	15	0.006434
14	11	0.004718
16	6	0.002573
17	6	0.002573
20	3	0.001287
15	2	0.000858
18	2	0.000858
19	2	0.000858
23	2	0.000858
28	1	0.000429
21	1	0.000429
22	1	0.000429
35	1	0.000429

Most of customers have not opened any new account in the last 6 months.Now, lets check loans defaulted in last six months.

In [76]:
```
delinquent_acct_counts = loandata['DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS'].value_counts()
delinquent_acct_counts_percent = loandata['DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS'].value_counts(normalize=True)*100

pd.DataFrame({'counts':delinquent_acct_counts,'delinquent_acct_counts':delinquent_acct_counts_percent})
```

Out[76]:

	counts	delinquent_acct_counts
0	214959	92.196145
1	14941	6.408211
2	2470	1.059386
3	537	0.230320
4	138	0.059188

	counts	delinquent_acct_counts
5	58	0.024876
6	20	0.008578
7	13	0.005576
8	7	0.003002
12	3	0.001287
11	3	0.001287
10	2	0.000858
9	2	0.000858
20	1	0.000429

We can obseved that can see that 92% of customers are not defaulted in last six months and 8% of customers are deafulted at least once or more than once in last 6 month.

Perform Model Building and Predict

In [77]:
```
#Perform logistic regression modelling, predict the outcome for the test data, and validate the results using the confu
```

In [78]:
```
#dropping unnecessary columns
# MobileNo_Avl_Flag - All values are 1
# Date_of_Birth , DisbursalDate -  Already used to compute age
# PERFORM_CNS_SCORE_DESCRIPTION - Score is already in dataset

loandata = loandata.drop(['MobileNo_Avl_Flag','Date_of_Birth','AVERAGE_ACCT_AGE','DisbursalDate','PERFORM_CNS_SCORE_DES
```

In [79]:
```
loandata_new = pd.get_dummies(loandata,drop_first=True)
print(loandata_new.columns)
```

```
Index(['UniqueID', 'disbursed_amount', 'asset_cost', 'ltv', 'branch_id',
       'supplier_id', 'manufacturer_id', 'Current_pincode_ID', 'State_ID',
       'Employee_code_ID', 'Aadhar_flag', 'PAN_flag', 'VoterID_flag',
       'Driving_flag', 'Passport_flag', 'PERFORM_CNS_SCORE', 'PRI_NO_OF_ACCTS',
       'PRI_ACTIVE_ACCTS', 'PRI_OVERDUE_ACCTS', 'PRI_CURRENT_BALANCE',
       'PRI_SANCTIONED_AMOUNT', 'PRI_DISBURSED_AMOUNT', 'SEC_NO_OF_ACCTS',
       'SEC_ACTIVE_ACCTS', 'SEC_OVERDUE_ACCTS', 'SEC_CURRENT_BALANCE',
       'SEC_SANCTIONED_AMOUNT', 'SEC_DISBURSED_AMOUNT', 'PRIMARY_INSTAL_AMT',
       'SEC_INSTAL_AMT', 'NEW_ACCTS_IN_LAST_SIX_MONTHS',
```

```
        'DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS', 'CREDIT_HISTORY_LENGTH',
        'NO_OF_INQUIRIES', 'loan_default', 'age',
        'Employment_Type_Self employed'],
       dtype='object')
```

In [80]:
```
#train test split
X = loandata_new.drop('loan_default',axis=1)
y = loandata_new['loan_default']
```

In [81]:
```
X.shape
```

Out[81]:
```
(233154, 36)
```

In [82]:
```
y.shape
```

Out[82]:
```
(233154,)
```

In [83]:
```
from sklearn.preprocessing import StandardScaler
```

In [84]:
```
sc = StandardScaler()
```

In [85]:
```
sc.fit(X)
```

Out[85]:
```
StandardScaler()
```

In [86]:
```
Xt_z = pd.DataFrame(sc.transform(X), columns =X.columns)
```

In [87]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(Xt_z, y, stratify=y, random_state=12)
```

In [88]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
fit=lr.fit(X_train, y_train)
```

In [89]:
```
y_pred= lr.predict(X_test)
```

In [90]:
```
pred_vs_actual_outcome = pd.crosstab(index = y_pred, columns = y_test)
pred_vs_actual_outcome
```

Out[90]:

loan_default	0	1

loan_default	0	1
row_0		
0	45545	12584
1	91	69

In [91]:
```
from sklearn.metrics import confusion_matrix
from sklearn import metrics
print(confusion_matrix(y_pred,y_test))
```

```
[[45545 12584]
 [   91    69]]
```

In [92]:
```
cm = confusion_matrix(y_pred,y_test)
```

In [93]:
```
#accuracy = (TN+TP)/(ALL)
accuracy_lr = (cm[0,0]+cm[1,1])/(cm[0,0]+cm[0,1]+cm[1,0]+cm[1,1])*100
print('accuracy' ,accuracy_lr)

#precision = (TP)/(TP+FP)
precision_lr = (cm[1,1])/(cm[1,1]+cm[1,0])*100
print('precision' ,precision_lr)


#recall or sensitivity(TPR) for class1 = (TP)/(TP+FN)
recall_lr_class_1 = (cm[1,1])/(cm[1,1]+cm[0,1])*100
print('class1-recall' ,recall_lr_class_1)

#recall or specificity(TNR) for class 0 = (TN)/(TN+FP)
recall_lr_class_0 = (cm[0,0])/(cm[0,0]+cm[0,1])*100
print('class0-recall' ,recall_lr_class_0)

#F1_Score or Harmonic mean(HM) of precision and recall  = 2*precision*recall/(precision + recall)
F1_Score_lr = (2*precision_lr*recall_lr_class_1*recall_lr_class_0)/(precision_lr+recall_lr_class_1+recall_lr_class_0)
print('F1_Score' ,F1_Score_lr)
```

```
accuracy 78.25490229717443
precision 43.125
class1-recall 0.5453252193155773
class0-recall 78.35159730943248
F1_Score 30.201233269920934
```

From above values, we can observe that accuracy of the model is 78% with precision that is the repeatatibilty of the predcited results is 43%. Recall or sensistvity of the model for class 1 is 0.54 and for class 0 is 78.35%. The model is giving very good sensitivty for class 0. The F1 score of the model is 30.20% and is too low to accpet the model's prediction.Now, export the data to excel for further vizualiation in Tableau.

In [94]:
```
loandata_output = loandata_new
```

We can add and drop variables and can focus on selected variables for vizualiation in Tableau.

In [95]:
```
loandata_output = pd.concat([loandata_output,importdata['PERFORM_CNS.SCORE.DESCRIPTION']], axis = 1)
```

In [96]:
```
loandata_output = loandata_output.drop(['asset_cost','ltv','Employee_code_ID','PRI_ACTIVE_ACCTS',
                                        'PRI_OVERDUE_ACCTS','PRI_CURRENT_BALANCE','SEC_ACTIVE_ACCTS','SEC_OVERDUE_ACCTS
                                       'PRIMARY_INSTAL_AMT','SEC_INSTAL_AMT'],axis=1)
```

In [97]:
```
loandata_output.to_excel('/Users/priya/Pravat/Simplilearn Data Analytics/Class 5/project/project 2/loandata_output.xlsx
```