Project on observation of factors affecting loan defaulters and model creation to predict potential defaulters

In [313…

```
import numpy as np;
import pandas as pd;
import openpyxl;
import matplotlib.pyplot as plt;
import seaborn as sns;
from scipy.stats import spearmanr;
from sklearn.linear_model import LogisticRegression;
from sklearn.model_selection import train_test_split,RandomizedSearchCV,GridSearchCV;
from sklearn.preprocessing import Normalizer;
from sklearn.metrics import confusion_matrix;
```

Week 1: Importing, Understanding, and Inspecting Data

In [2]:

```
loan_df=pd.read_excel("data.xlsx");
```

1. Perform preliminary data inspection and report the findings as the structure of the data, missing values, duplicates, etc.

In [3]:

```
loan_df
```

Out[3]:

	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	Salaried	...
1	417566	53278	61360	89.63	67	22807	45	1497	1985-08-24	Self employed	...
2	539055	52378	60300	88.39	67	22807	45	1495	1977-12-09	Self employed	...
3	529269	46349	61500	76.42	67	22807	45	1502	1988-06-01	Salaried	...
4	563215	43594	78256	57.50	67	22744	86	1499	1994-07-14	Self employed	...
...	...	...	...	...	...	...	...	...	...	...	...

	UniqueID	disbursed_amount	asset_cost	ltv	branch_id	supplier_id	manufacturer_id	Current_pincode_ID	Date.of.Birth	Employment.Type	...
233149	561031	57759	76350	77.28	5	22289	51	3326	1981-11-10	Self employed	...
233150	649600	55009	71200	78.72	138	17408	51	3385	1992-10-15	Self employed	...
233151	603445	58513	68000	88.24	135	23313	45	1797	1981-12-19	Self employed	...
233152	442948	22824	40458	61.79	160	16212	48	96	1989-07-31	Self employed	...
233153	545300	35299	72698	52.27	3	14573	45	17	1968-08-01	Self employed	...

233154 rows × 41 columns

In [4]:

```
loan_df.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]:

```
len(loan_df['UniqueID'])
```

Out[5]:

```
233154
```

In [6]:

```
len((loan_df['UniqueID']).unique())
```

Out[6]:

```
233154
```

"Employment.Type" column has few missing values. There are no duplicate IDs

2. Variable names in the data may not be in accordance with the identifier naming in Python so, change the variable names accordingly

In [7]:

```
loan_df.columns
```

Out[7]:
```
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 [8]:
```
loan_df.columns=[val.replace('.','_') for val in loan_df.columns]
```

In [9]:
```
loan_df.columns
```

Out[9]:
```
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')
```

3. The presented data might also contain some missing values therefore, exploration will also lead to devising strategies to fill in the missing values while exploring the data

In [10]:
```
len(loan_df[loan_df.Employment_Type.isna()])
```

Out[10]:
```
7661
```

In [11]:
```
loan_df.Employment_Type.value_counts()
```

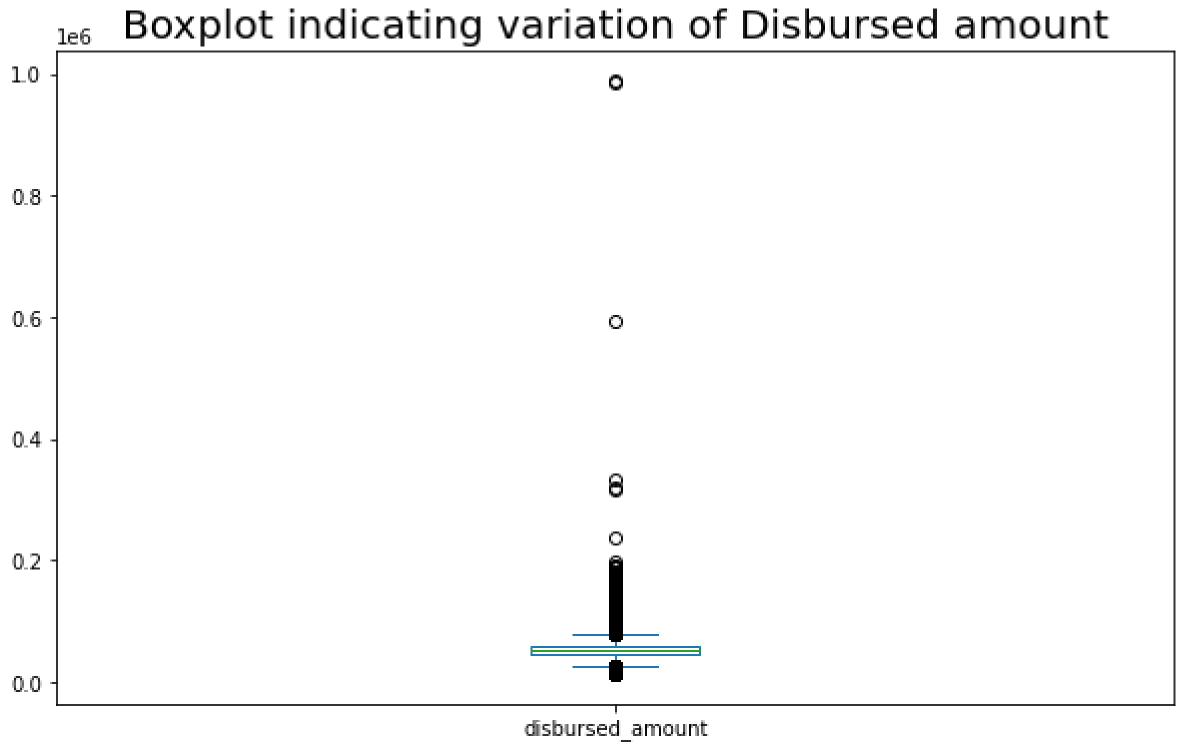
Out[11]:
```
Self employed    127635
Salaried          97858
Name: Employment_Type, dtype: int64
```

4. Provide the statistical description of the quantitative data variables

In [12]:
```
loan_df.disbursed_amount
```

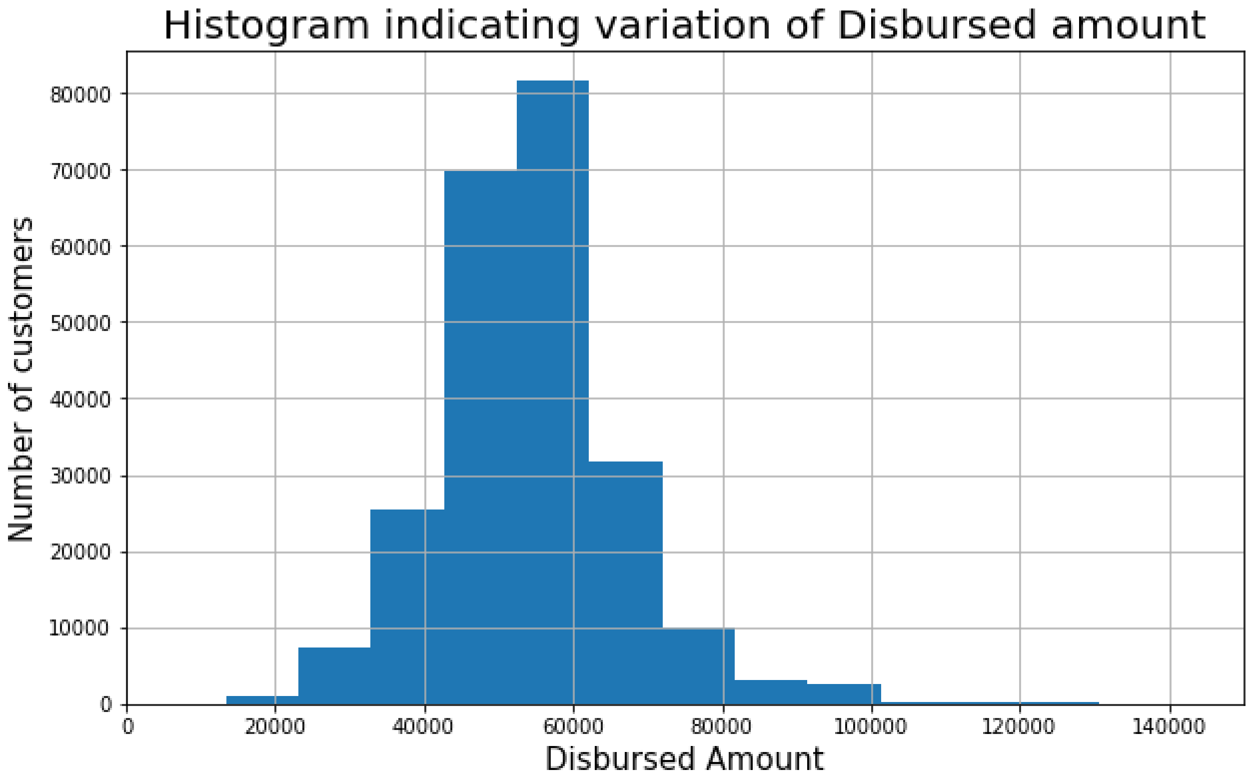
Out[12]:
```
0         50578
1         53278
2         52378
3         46349
4         43594
          ...
233149    57759
233150    55009
233151    58513
233152    22824
233153    35299
Name: disbursed_amount, Length: 233154, dtype: int64
```

In [13]:
```
fig1,ax1=plt.subplots(1,1,figsize=(10,6));
loan_df.disbursed_amount.plot.box(ax=ax1);
ax1.set_title("Boxplot indicating variation of Disbursed amount",size=20);
```

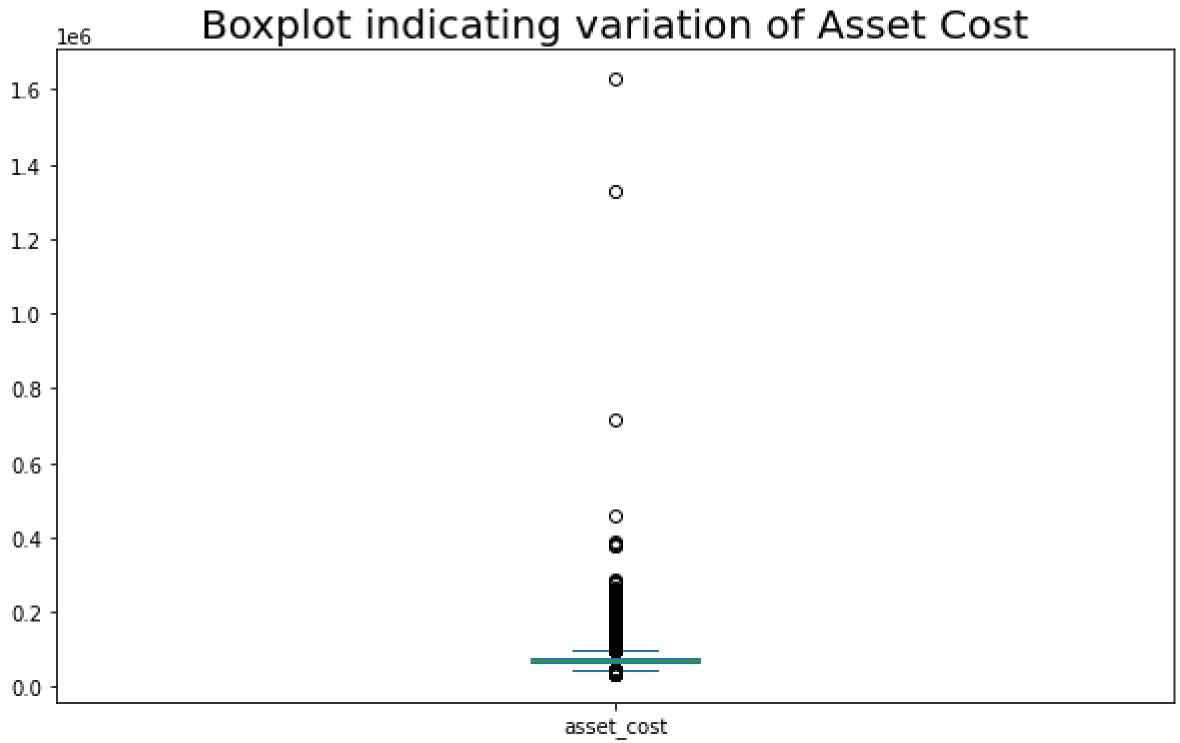


In [14]:

```
fig2,ax2=plt.subplots(1,1,figsize=(10,6));
loan_df.disbursed_amount.hist(ax=ax2,bins=100);
ax2.set_xlim([0,150000]);
ax2.set_xlabel("Disbursed Amount",size=15);
ax2.set_ylabel("Number of customers",size=15);
ax2.set_title("Histogram indicating variation of Disbursed amount",size=20);
```

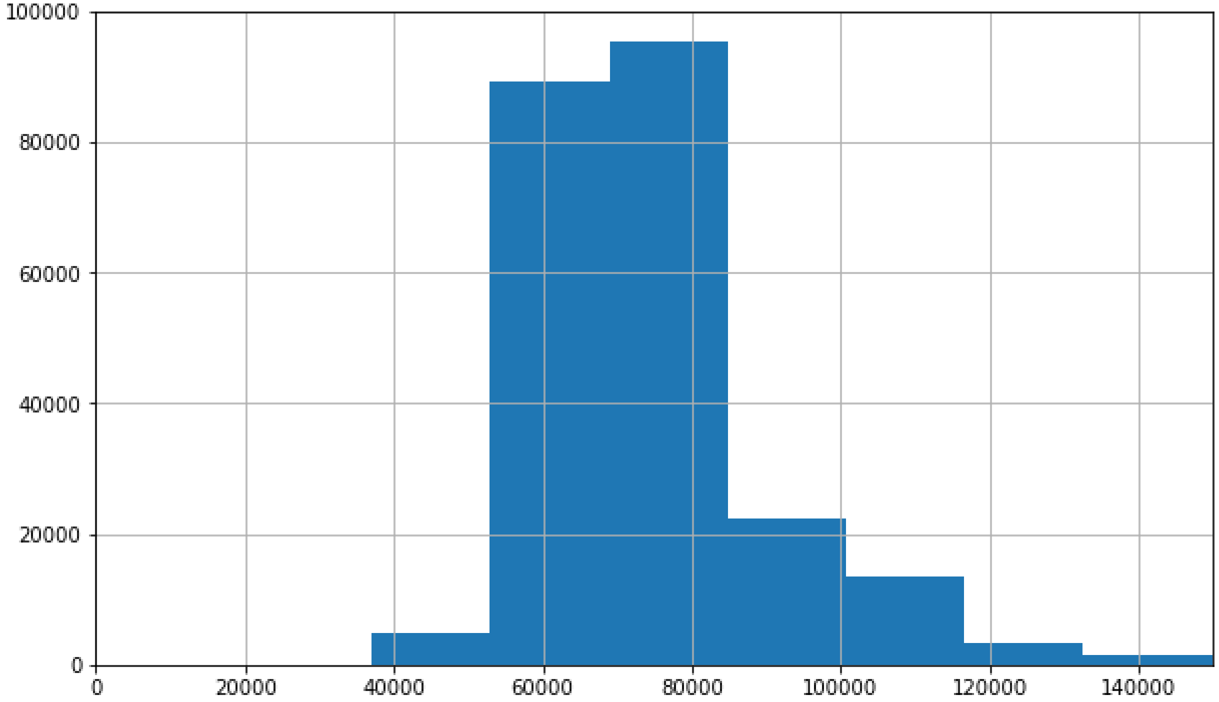


In [15]:
```
fig3,ax3=plt.subplots(1,1,figsize=(10,6));
loan_df.asset_cost.plot.box(ax=ax3);
ax3.set_title("Boxplot indicating variation of Asset Cost",size=20);
```

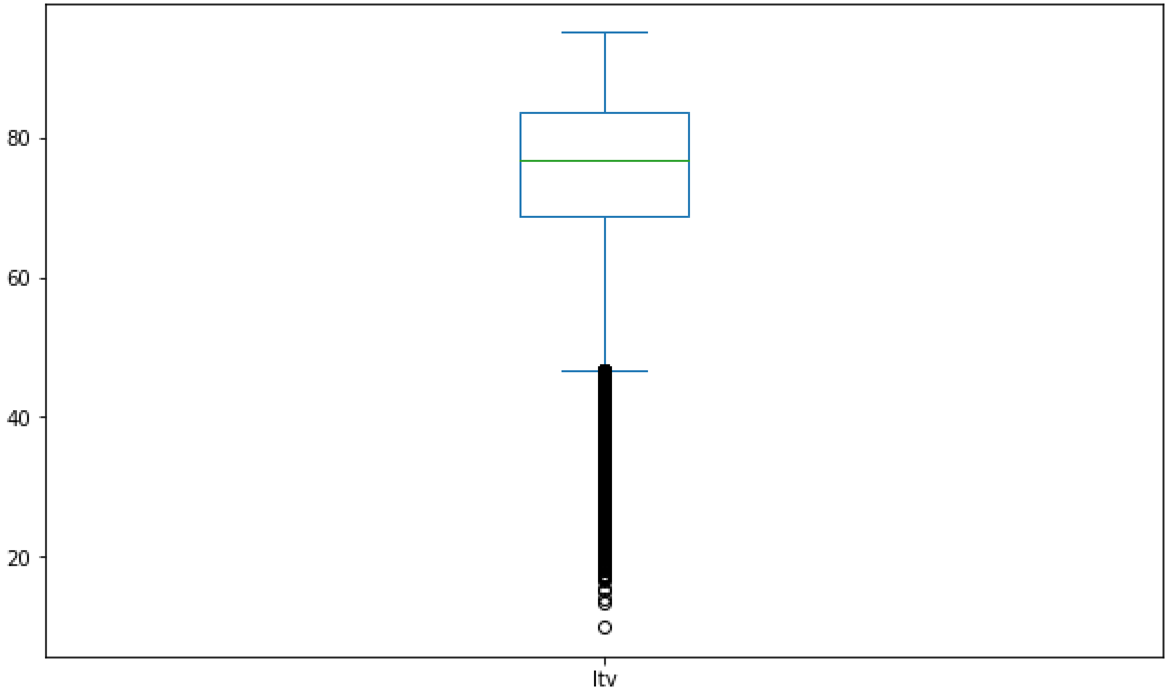


In [16]:

```
fig4,ax4=plt.subplots(1,1,figsize=(10,6));
loan_df.asset_cost.hist(ax=ax4,bins=100);
ax4.set_xlim([0,150000]);
```



In [17]:
```
fig5,ax5=plt.subplots(1,1,figsize=(10,6));
loan_df.ltv.plot.box(ax=ax5);
```



In [18]:
```
loan_df.Date_of_Birth
```

Out[18]:
```
0         1984-01-01
1         1985-08-24
2         1977-12-09
3         1988-06-01
4         1994-07-14
             ...
233149    1981-11-10
233150    1992-10-15
233151    1981-12-19
233152    1989-07-31
233153    1968-08-01
Name: Date_of_Birth, Length: 233154, dtype: datetime64[ns]
```

In [19]:
```
import datetime as dt
```

In [20]:
```
dt.date.today()
```

Out[20]:

```
datetime.date(2022, 11, 17)
```

In [21]:

```
loan_df['age']=[int(((dt.datetime.today()-i).days)/365.25) for i in  loan_df.Date_of_Birth];
```

In [22]:

```
loan_df
```

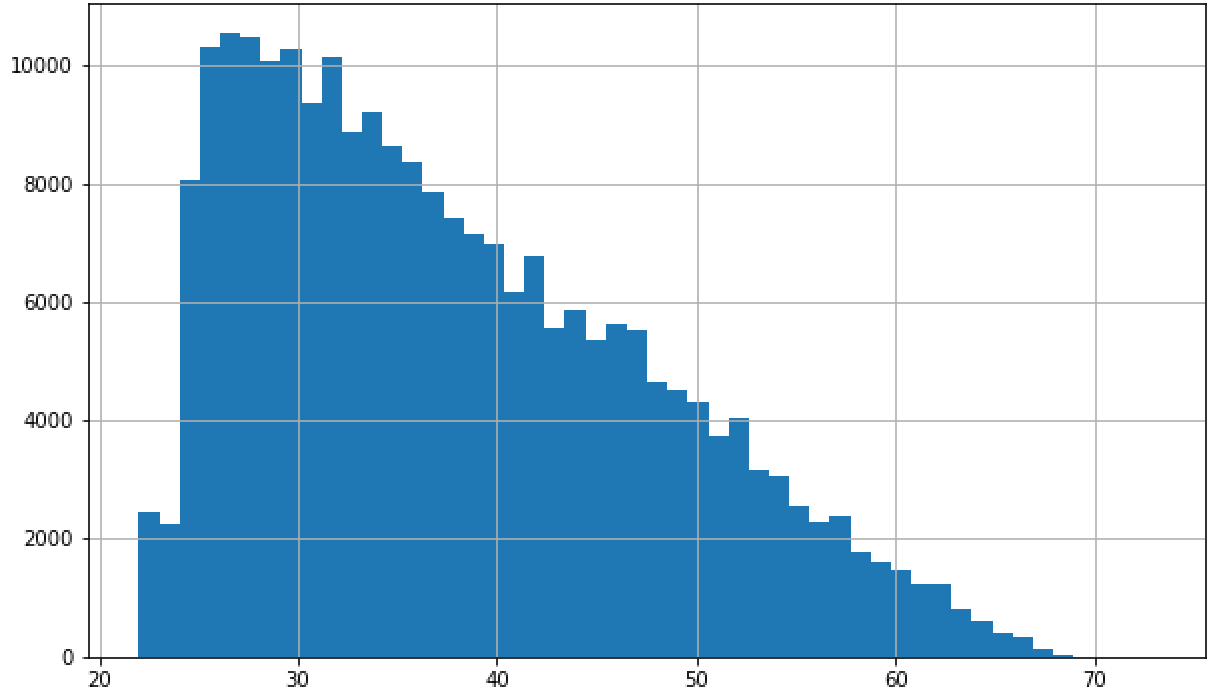
Out[22]:

	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	Salaried	...
1	417566	53278	61360	89.63	67	22807	45	1497	1985-08-24	Self employed	...
2	539055	52378	60300	88.39	67	22807	45	1495	1977-12-09	Self employed	...
3	529269	46349	61500	76.42	67	22807	45	1502	1988-06-01	Salaried	...
4	563215	43594	78256	57.50	67	22744	86	1499	1994-07-14	Self employed	...
...	...	...	...	...	...	...	...	...	...	...	...
233149	561031	57759	76350	77.28	5	22289	51	3326	1981-11-10	Self employed	...
233150	649600	55009	71200	78.72	138	17408	51	3385	1992-10-15	Self employed	...
233151	603445	58513	68000	88.24	135	23313	45	1797	1981-12-19	Self employed	...
233152	442948	22824	40458	61.79	160	16212	48	96	1989-07-31	Self employed	...
233153	545300	35299	72698	52.27	3	14573	45	17	1968-08-01	Self employed	...

233154 rows × 42 columns

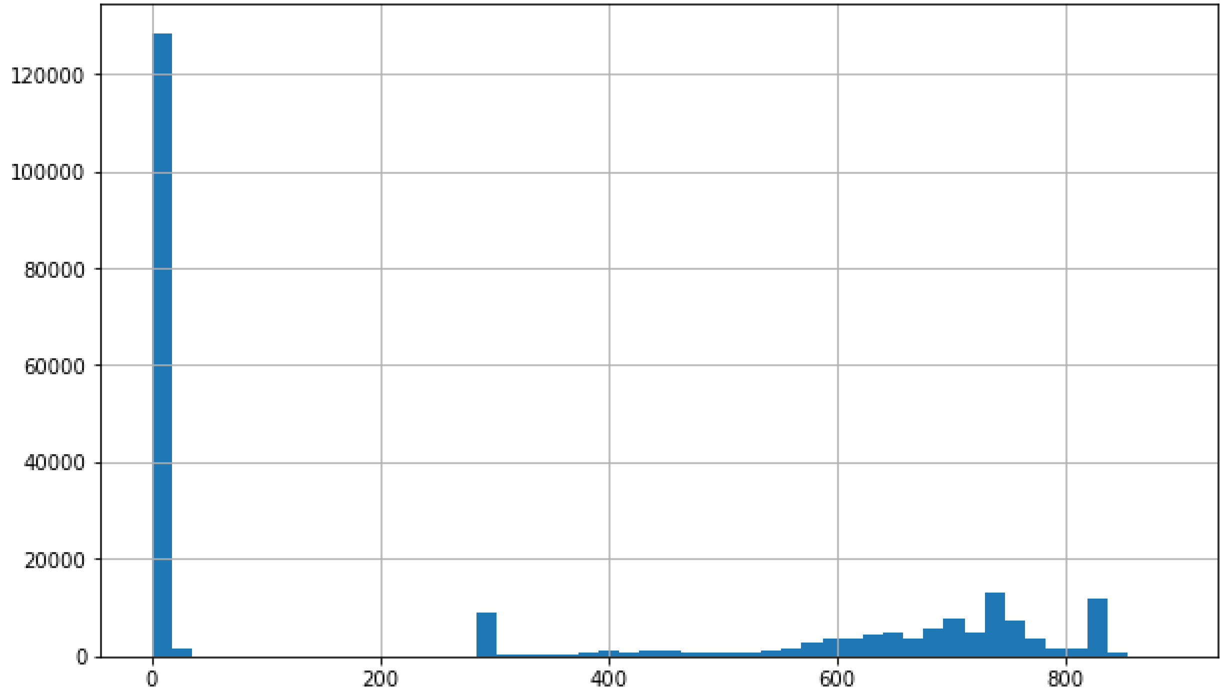
In [23]:

```
fig5,ax5=plt.subplots(1,1,figsize=(10,6));
loan_df['age'].hist(ax=ax5,bins=50);
```



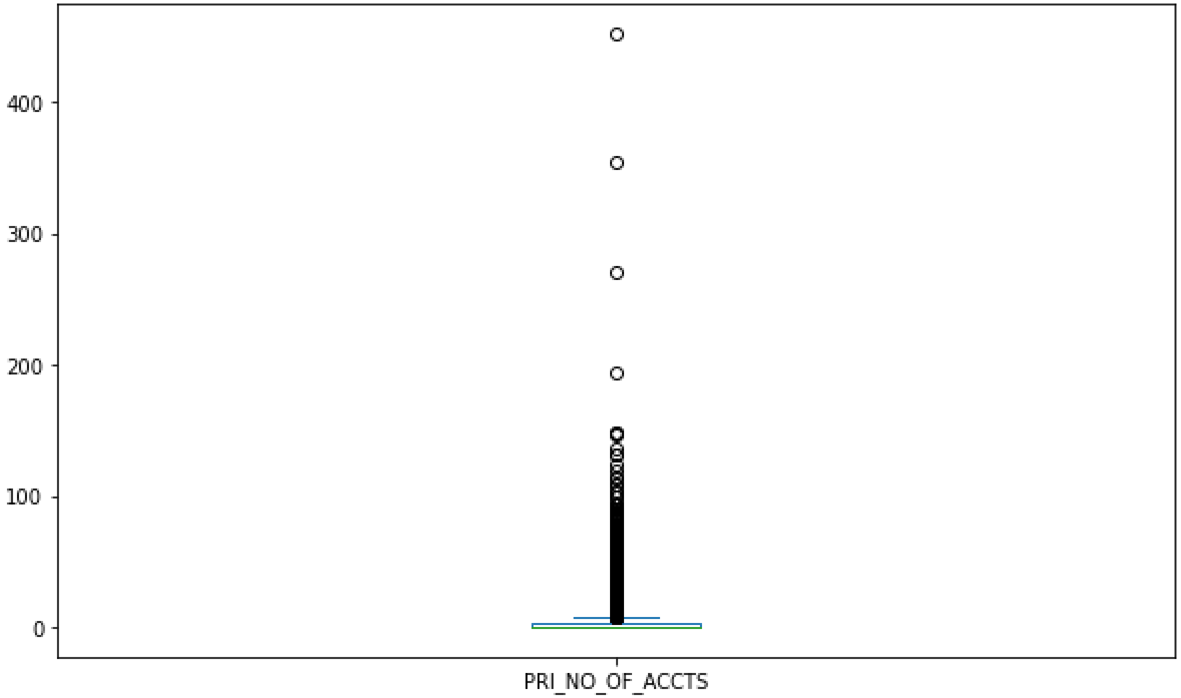
In [24]:

```
fig6,ax6=plt.subplots(1,1,figsize=(10,6));
loan_df.PERFORM_CNS_SCORE.hist(ax=ax6,bins=50);
```

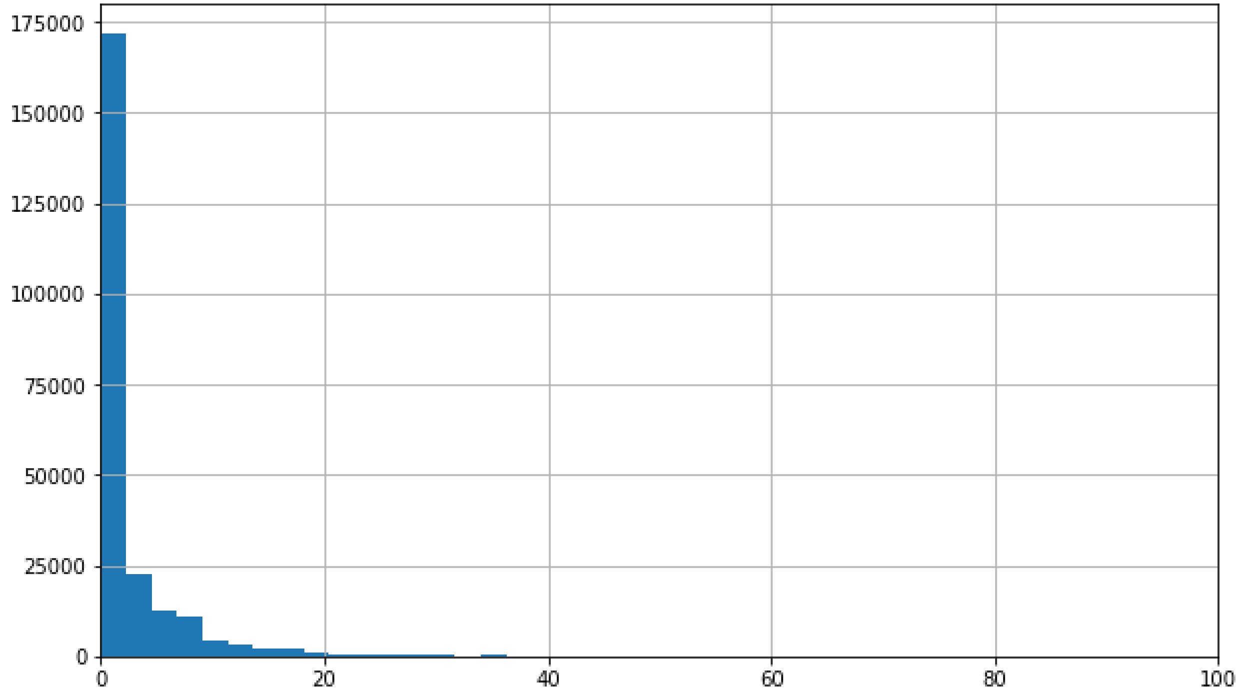


In [25]:

```
fig7,ax7=plt.subplots(1,1,figsize=(10,6));
loan_df.PRI_NO_OF_ACCTS.plot.box(ax=ax7);
```



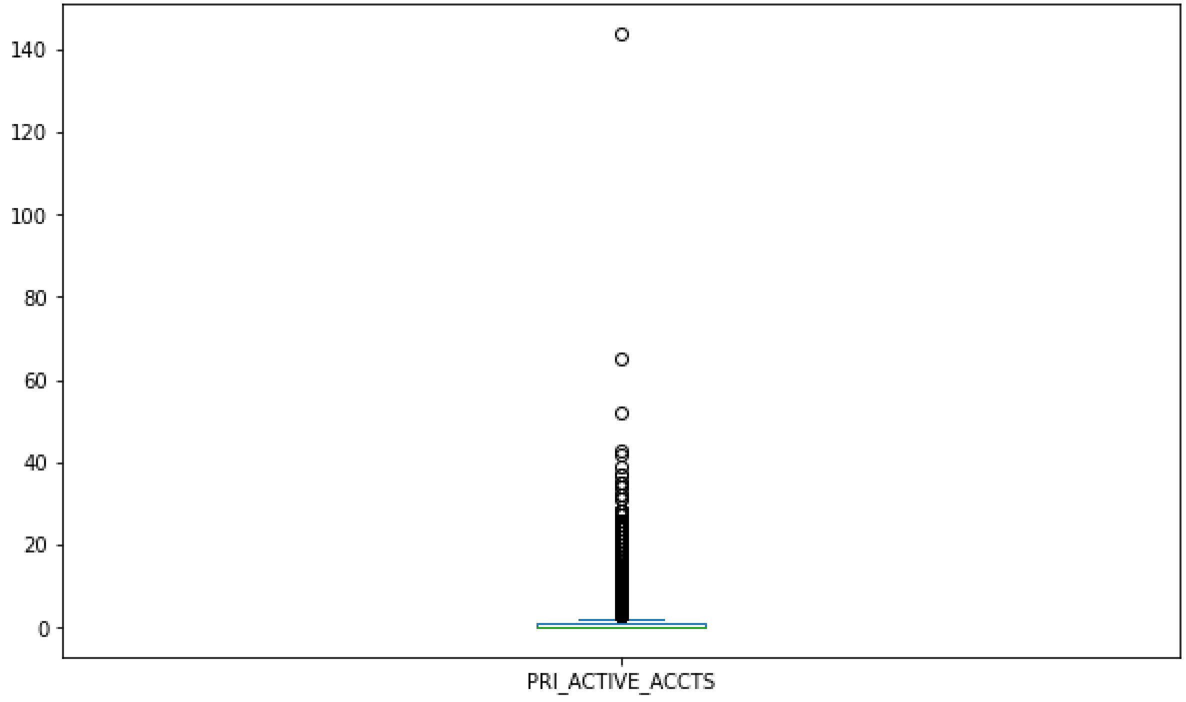
In [26]:
```
fig8,ax8=plt.subplots(1,1,figsize=(10,6));
loan_df.PRI_NO_OF_ACCTS.hist(ax=ax8,bins=200);
ax8.set_xlim([0,100]);
```



In [27]:
```
loan_df.PRI_NO_OF_ACCTS.value_counts()
```

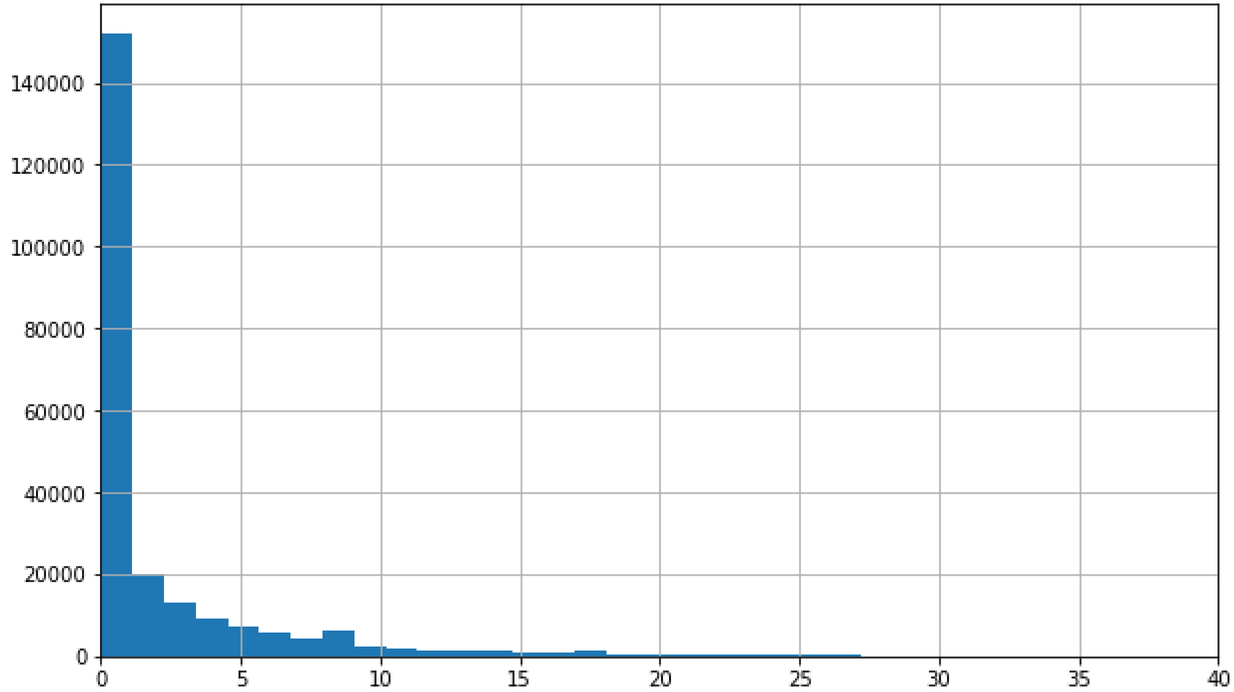
Out[27]:
```
0      116950
1       34978
2       19784
3       13015
4        9323
        ...  
85          1
131         1
124         1
453         1
194         1
Name: PRI_NO_OF_ACCTS, Length: 108, dtype: int64
```

In [28]:
```
fig9,ax9=plt.subplots(1,1,figsize=(10,6));
loan_df.PRI_ACTIVE_ACCTS.plot.box(ax=ax9);
```



In [29]:

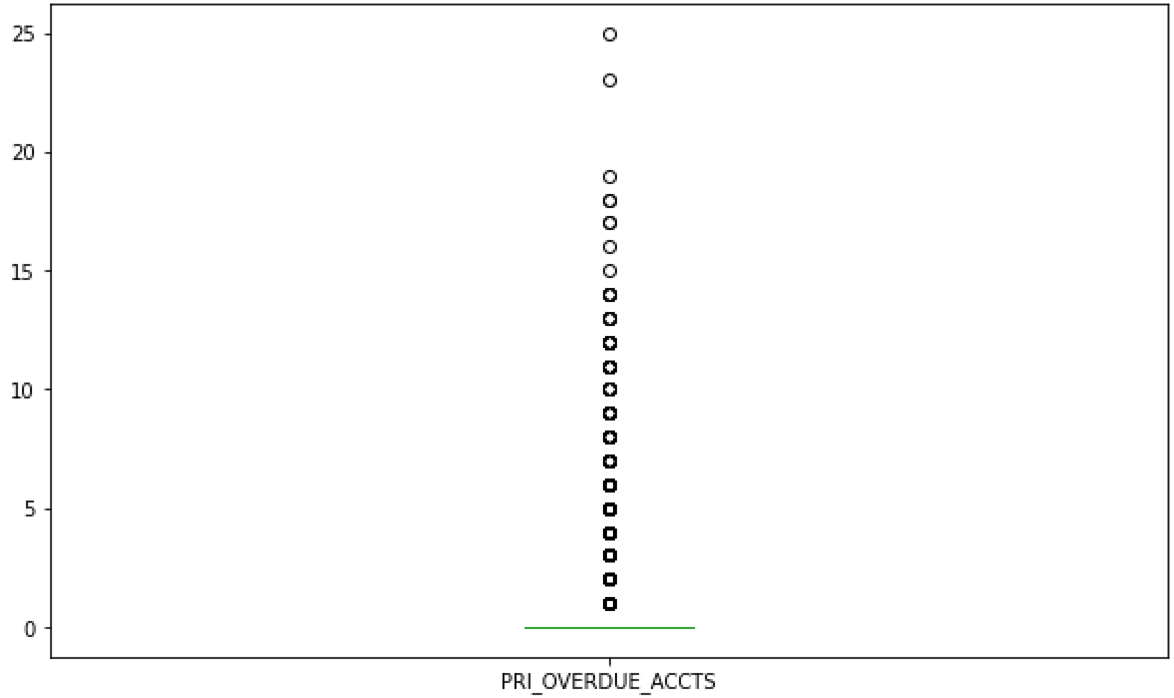
```
fig10,ax10=plt.subplots(1,1,figsize=(10,6));
loan_df.PRI_NO_OF_ACCTS.hist(ax=ax10,bins=400);
ax10.set_xlim([0,40]);
```



In [30]:
```
loan_df.PRI_NO_OF_ACCTS.value_counts()
```

Out[30]:
```
0      116950
1       34978
2       19784
3       13015
4        9323
        ...  
85          1
131         1
124         1
453         1
194         1
Name: PRI_NO_OF_ACCTS, Length: 108, dtype: int64
```

In [31]:
```
fig11,ax11=plt.subplots(1,1,figsize=(10,6));
loan_df.PRI_OVERDUE_ACCTS.plot.box(ax=ax11);
```



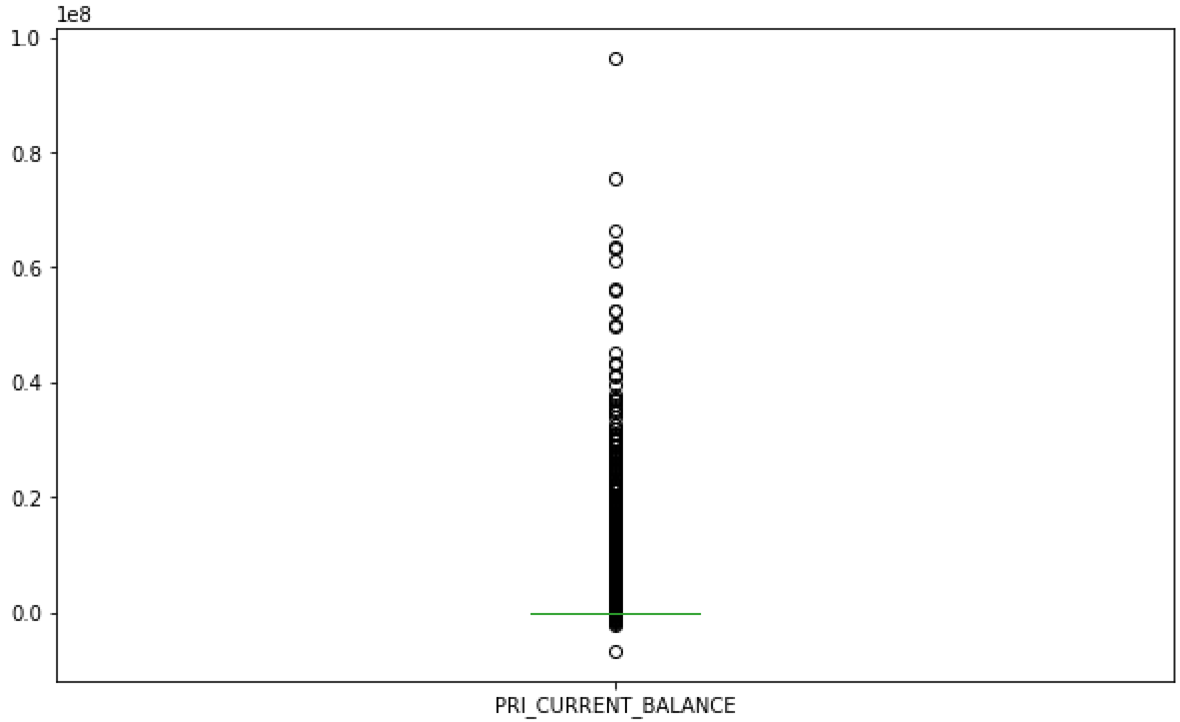
In [32]:
```
loan_df.PRI_OVERDUE_ACCTS.value_counts()
```

Out[32]:
```
0     206879
1      19970
2       4302
3       1202
4        404
5        166
6         96
7         38
8         27
9         25
11        12
12         8
10         6
14         5
13         5
18         2
17         2
23         1
```

```
19          1
15          1
16          1
25          1
Name: PRI_OVERDUE_ACCTS, dtype: int64
```

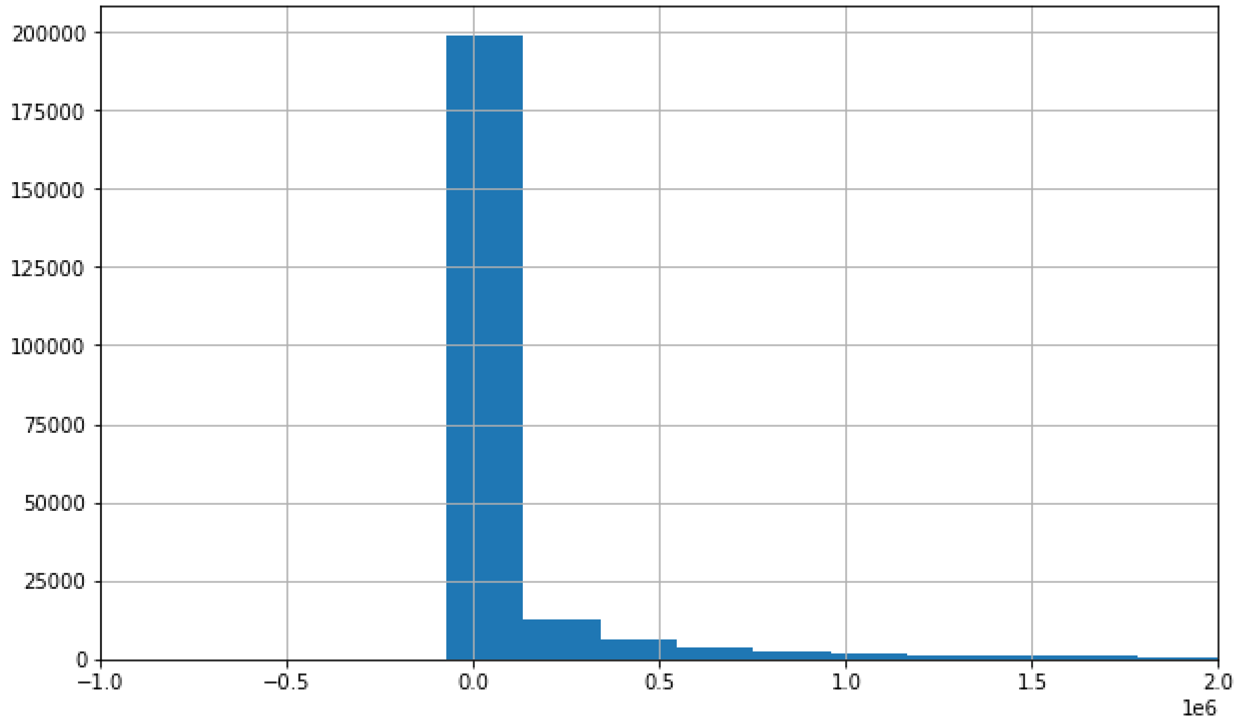
In [33]:

```
fig12,ax12=plt.subplots(1,1,figsize=(10,6));
loan_df.PRI_CURRENT_BALANCE.plot.box(ax=ax12);
```

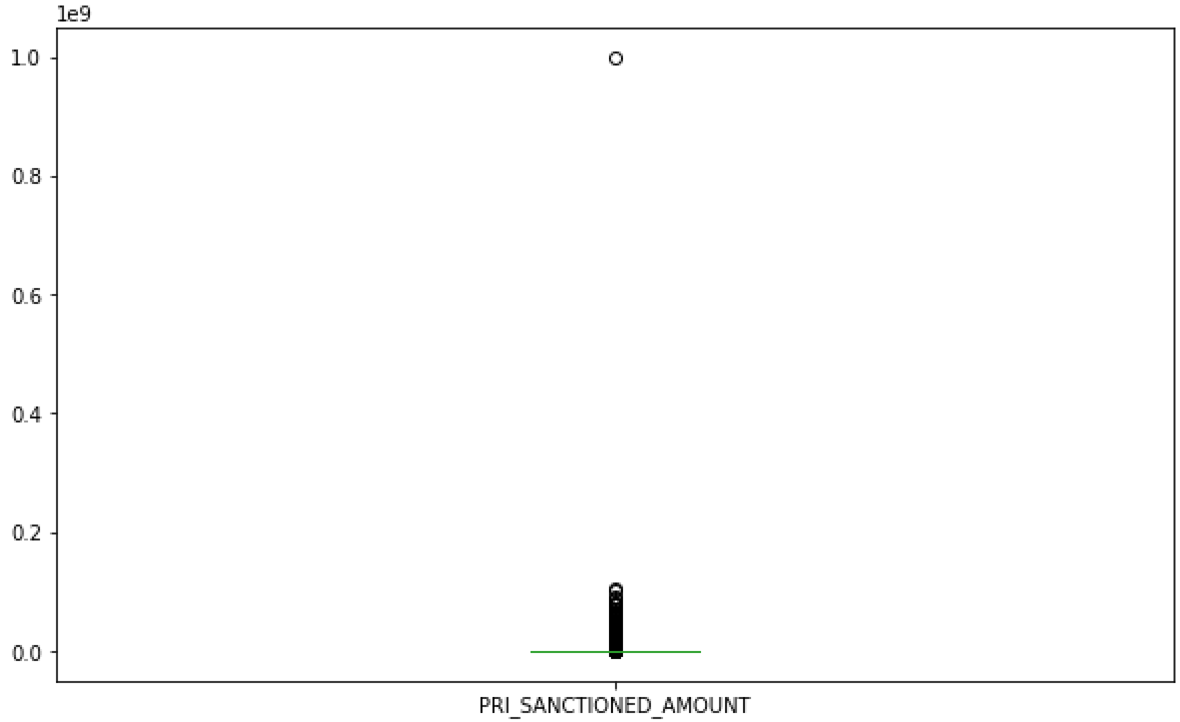


In [34]:

```
fig13,ax13=plt.subplots(1,1,figsize=(10,6));
loan_df.PRI_CURRENT_BALANCE.hist(ax=ax13,bins=500);
ax13.set_xlim([-1000000,2000000]);
```

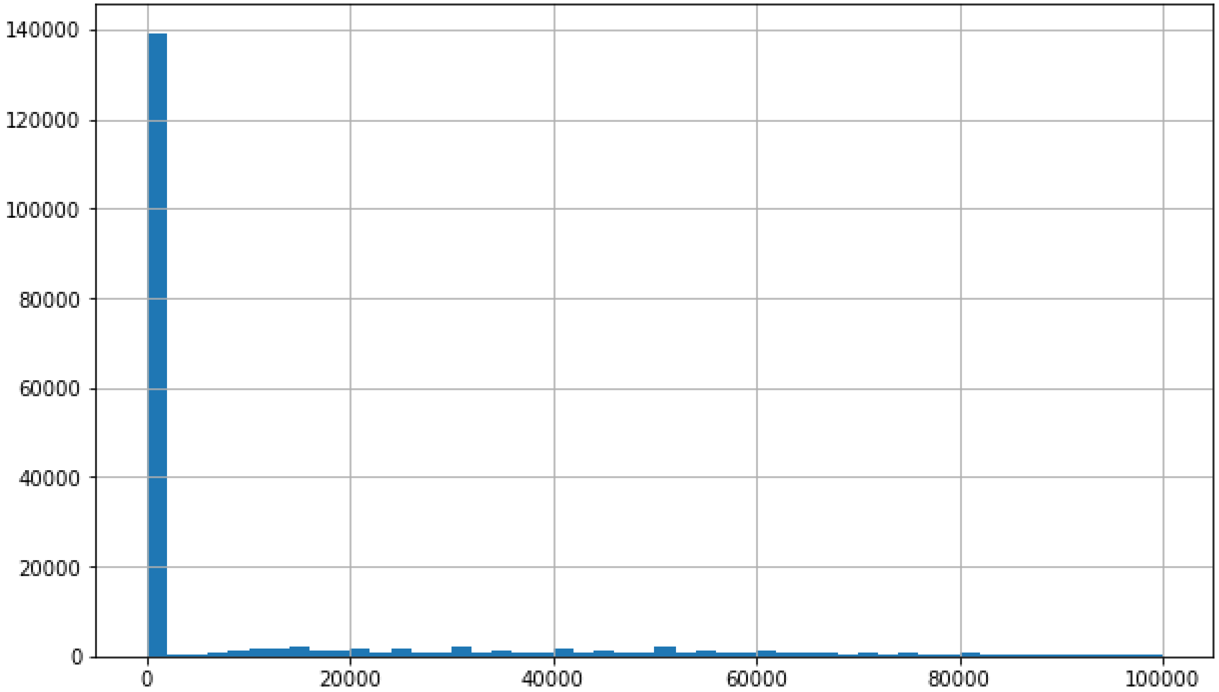


In [35]:
```
fig14,ax14=plt.subplots(1,1,figsize=(10,6));
loan_df.PRI_SANCTIONED_AMOUNT.plot.box(ax=ax14);
```

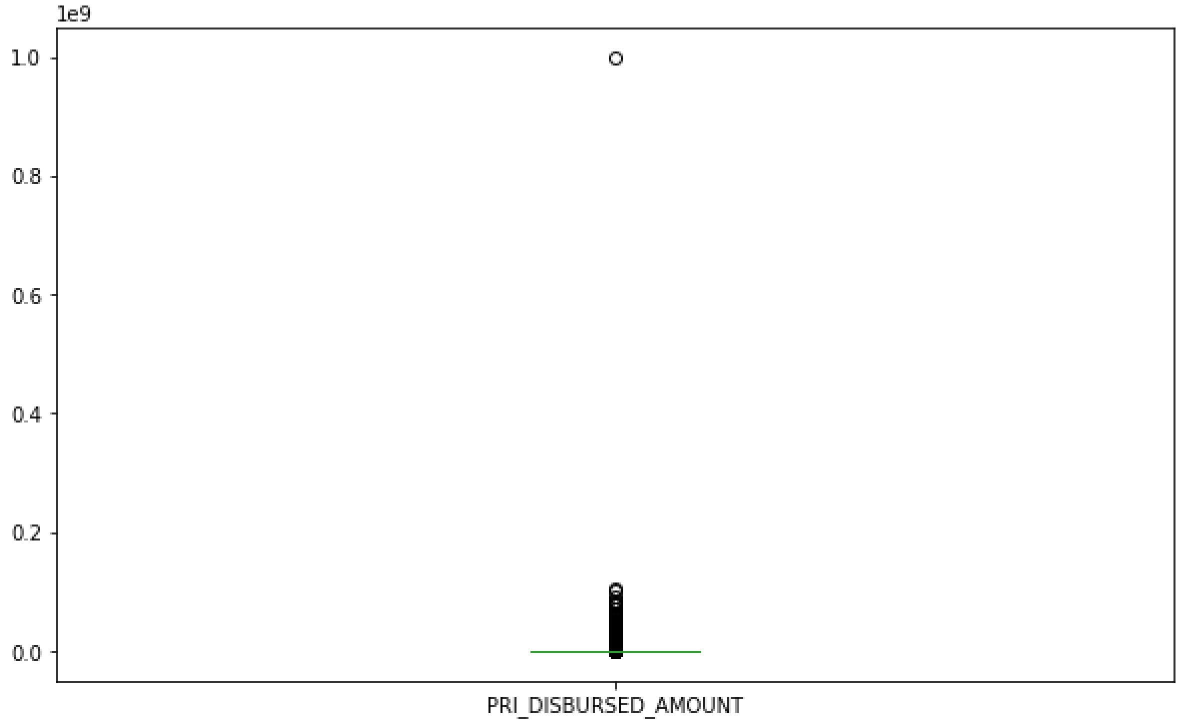


In [36]:

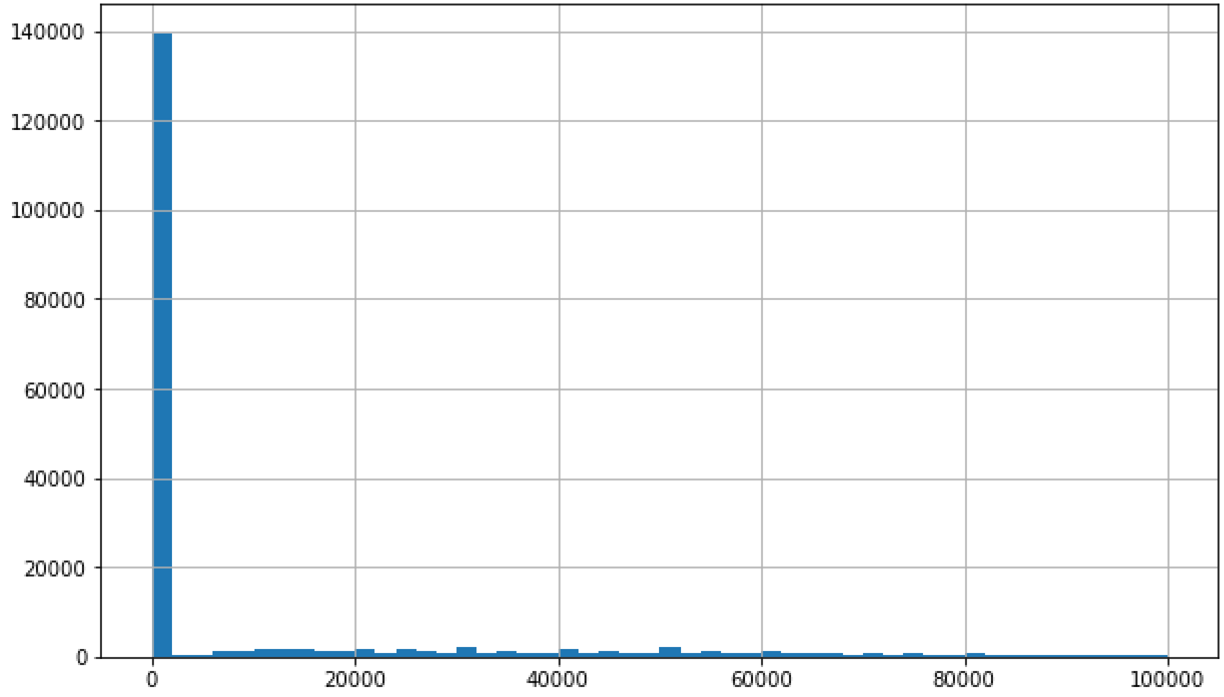
```
fig15,ax15=plt.subplots(1,1,figsize=(10,6));
loan_df.PRI_SANCTIONED_AMOUNT[loan_df.PRI_SANCTIONED_AMOUNT<100000].hist(ax=ax15,bins=50);
```



In [37]:
```
fig16,ax16=plt.subplots(1,1,figsize=(10,6));
loan_df.PRI_DISBURSED_AMOUNT.plot.box(ax=ax16);
```

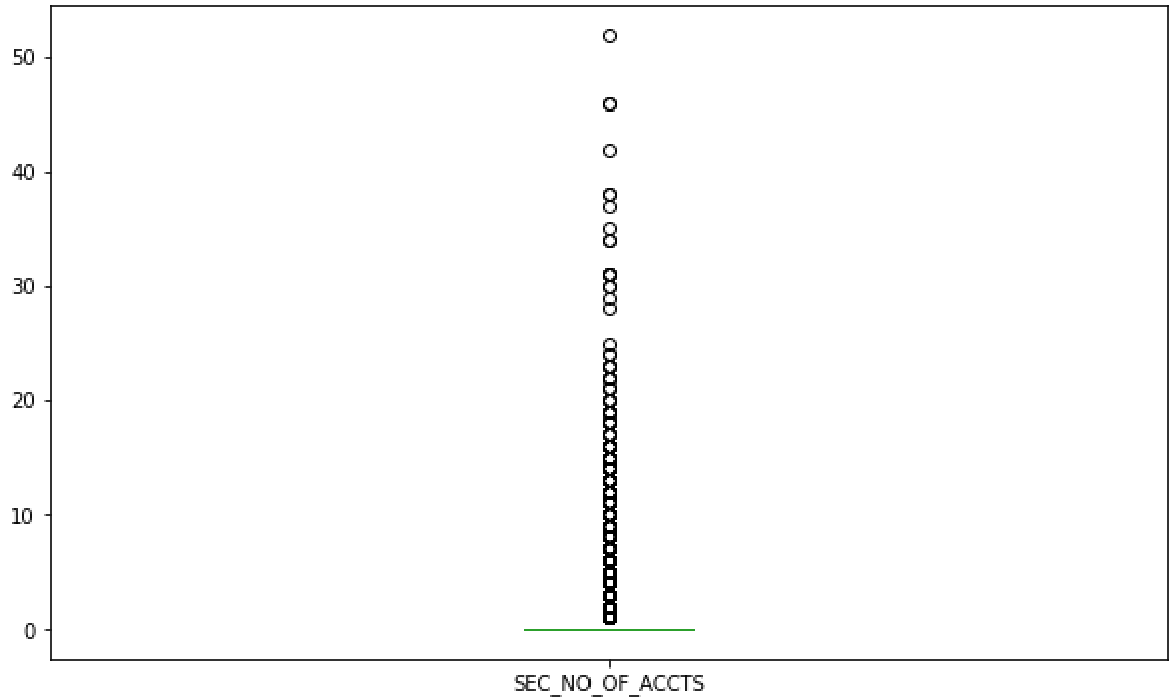


In [38]:
```
fig17,ax17=plt.subplots(1,1,figsize=(10,6));
loan_df.PRI_DISBURSED_AMOUNT[loan_df.PRI_DISBURSED_AMOUNT<100000].hist(ax=ax17,bins=50);
```



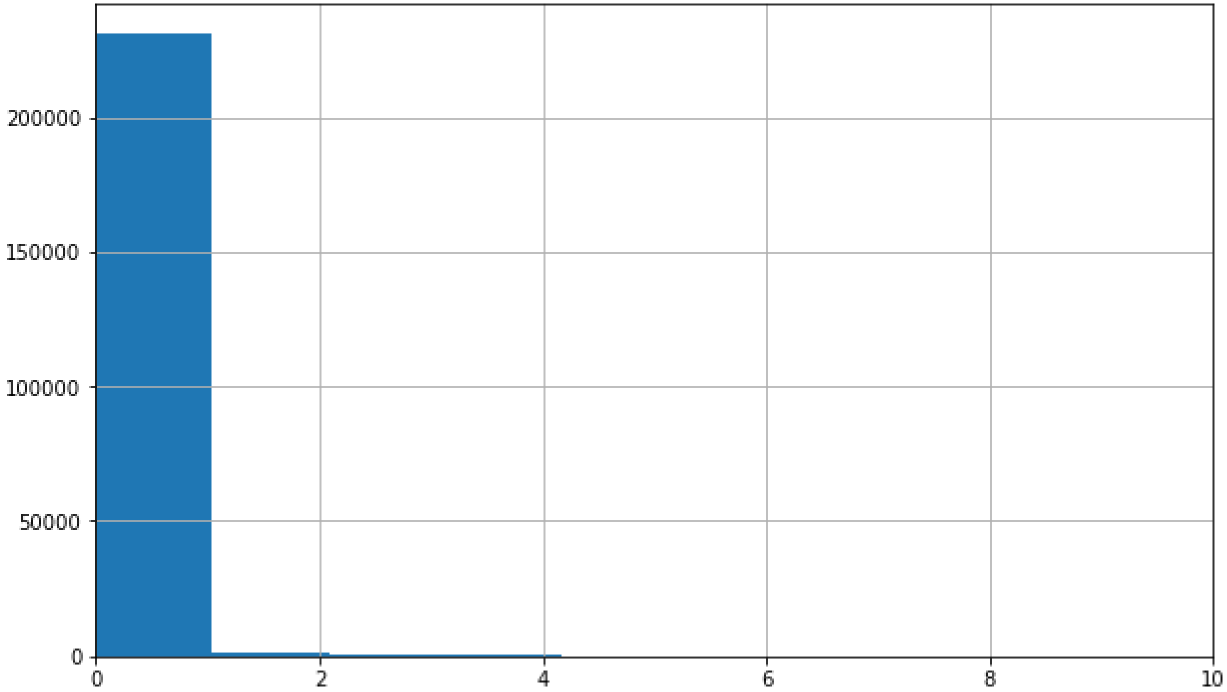
In [39]:

```
fig18,ax18=plt.subplots(1,1,figsize=(10,6));
loan_df.SEC_NO_OF_ACCTS.plot.box(ax=ax18);
```



In [40]:

```
fig19,ax19=plt.subplots(1,1,figsize=(10,6));
loan_df.SEC_NO_OF_ACCTS.hist(ax=ax19,bins=50);
ax19.set_xlim([0,10]);
```



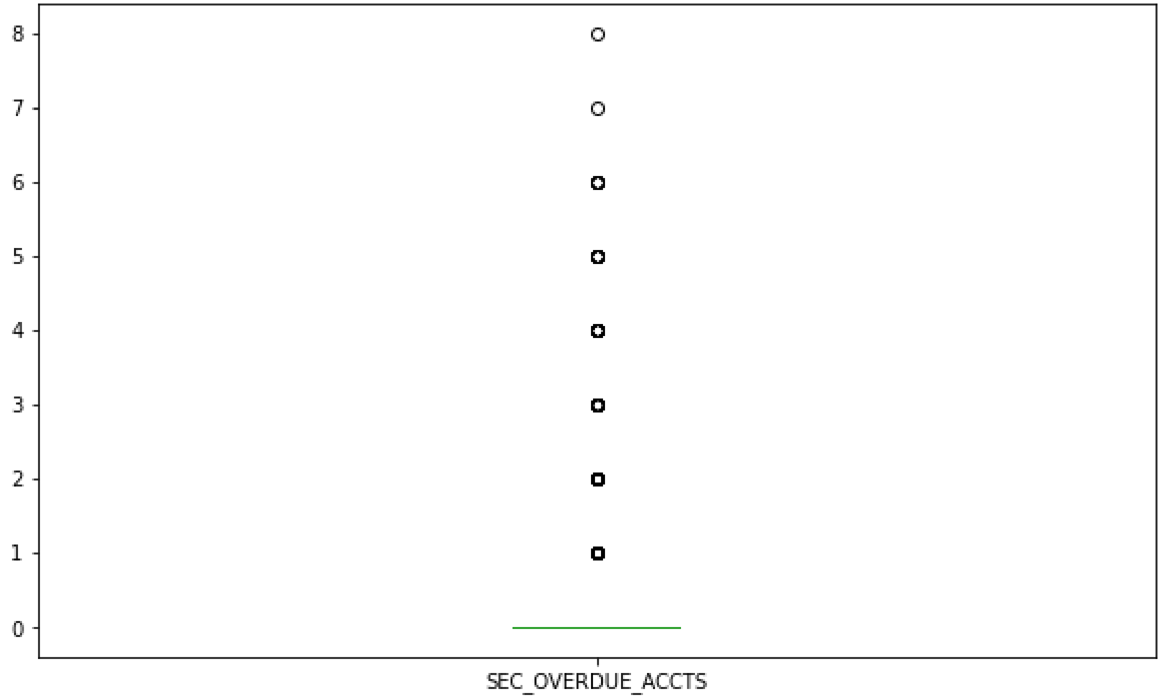
In [41]:
```
loan_df.SEC_NO_OF_ACCTS.value_counts()
```

Out[41]:
```
0     227289
1       3466
2       1036
3        444
4        292
5        148
6        119
7         75
8         68
9         38
10        35
11        29
13        17
12        13
14        11
16        11
15        10
18         6
```

```
19        6
17        5
23        4
31        4
22        4
20        4
21        3
46        2
24        2
34        2
30        2
38        2
35        1
25        1
28        1
37        1
42        1
52        1
29        1
Name: SEC_NO_OF_ACCTS, dtype: int64
```

In [42]:

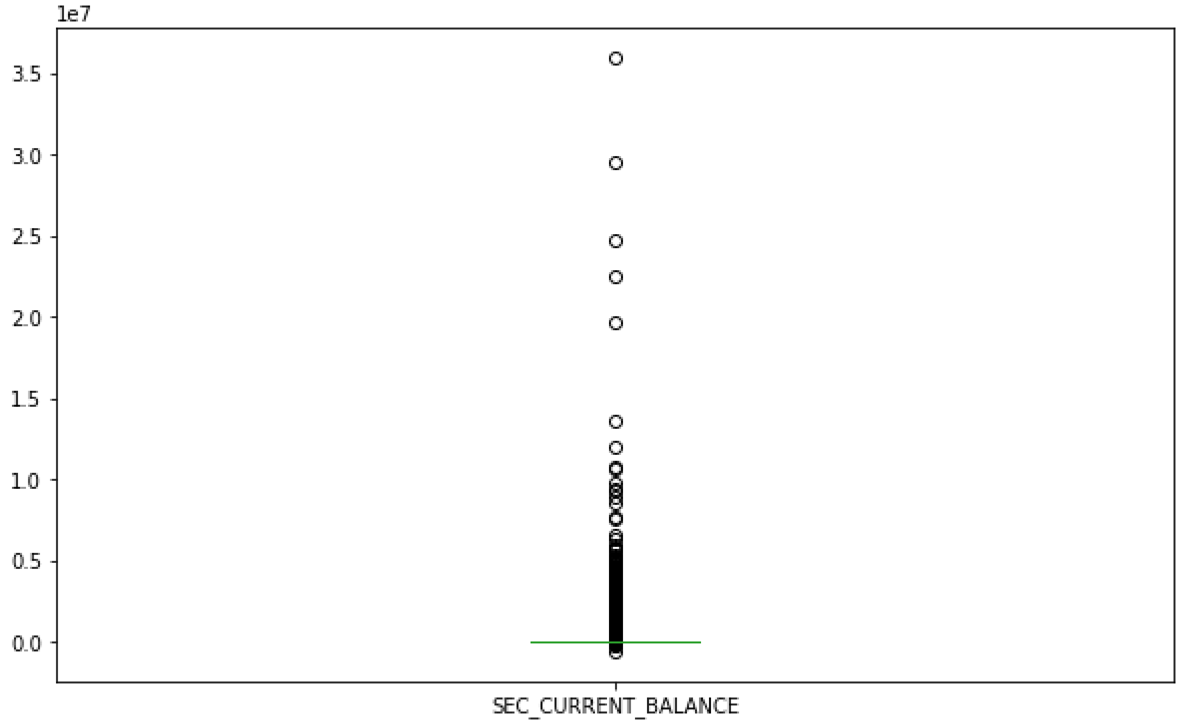
```
fig20,ax20=plt.subplots(1,1,figsize=(10,6));
loan_df.SEC_OVERDUE_ACCTS.plot.box(ax=ax20);
```



In [43]:
```
loan_df.SEC_OVERDUE_ACCTS.value_counts()
```

Out[43]:
```
0    231817
1      1129
2       126
3        47
4        19
5         8
6         6
7         1
8         1
Name: SEC_OVERDUE_ACCTS, dtype: int64
```

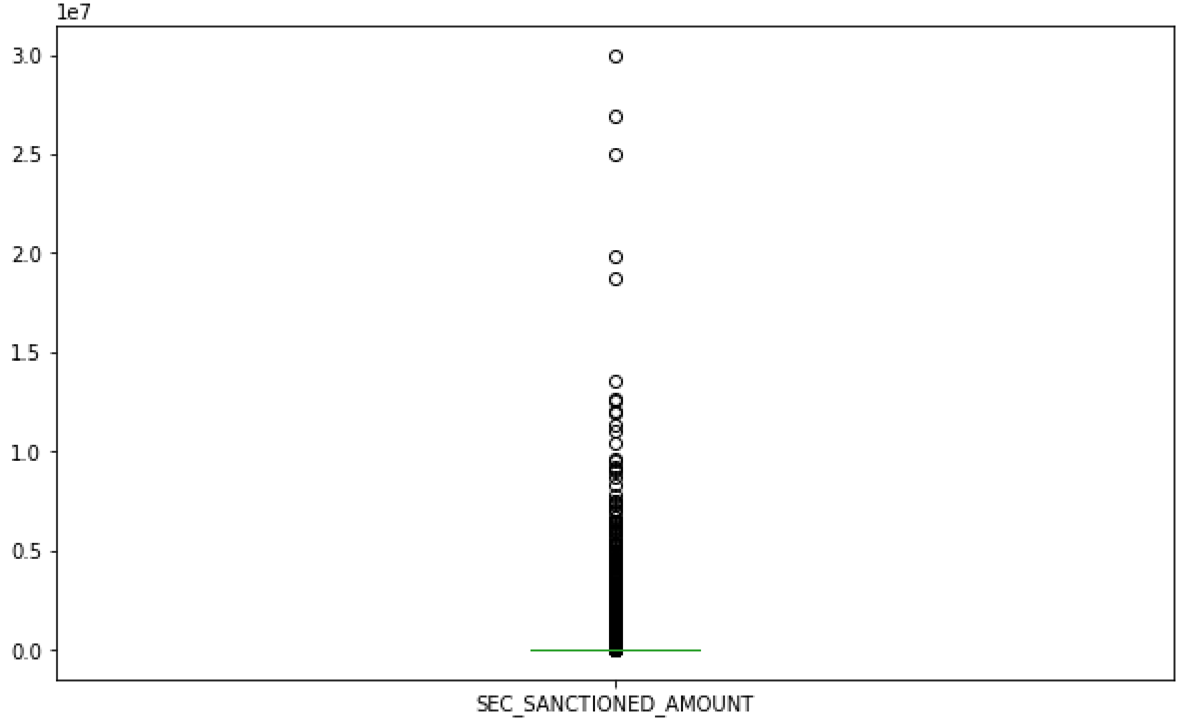
In [44]:
```
fig21,ax21=plt.subplots(1,1,figsize=(10,6));
loan_df.SEC_CURRENT_BALANCE.plot.box(ax=ax21);
```



In [45]:
```
loan_df.SEC_CURRENT_BALANCE.value_counts()
```

Out[45]:
```
0          229790
800            10
100             8
400             8
589             6
            ...
25920           1
4979            1
249287          1
1799            1
1119615         1
Name: SEC_CURRENT_BALANCE, Length: 3246, dtype: int64
```

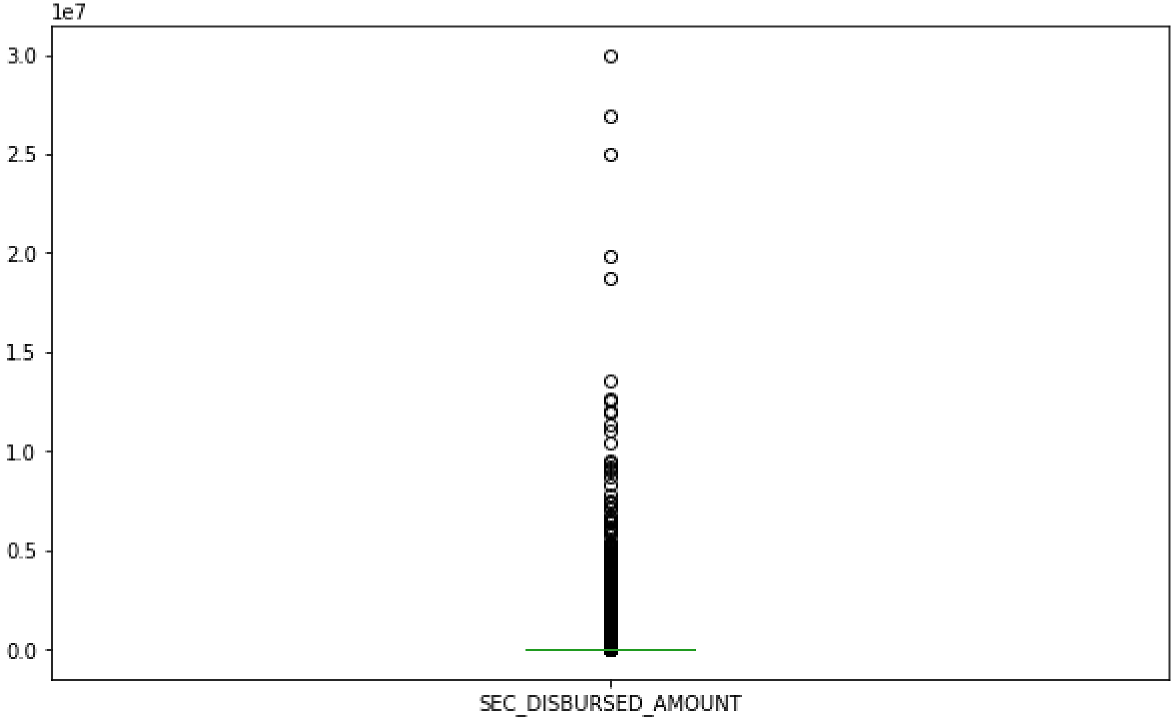
In [46]:
```
fig22,ax22=plt.subplots(1,1,figsize=(10,6));
loan_df.SEC_SANCTIONED_AMOUNT.plot.box(ax=ax22);
```



In [47]:
```
loan_df.SEC_SANCTIONED_AMOUNT.value_counts()
```

Out[47]:
```
0          229418
50000          83
100000         61
30000          44
40000          39
            ...  
14300           1
43225           1
295000          1
752933          1
360499          1
Name: SEC_SANCTIONED_AMOUNT, Length: 2223, dtype: int64
```

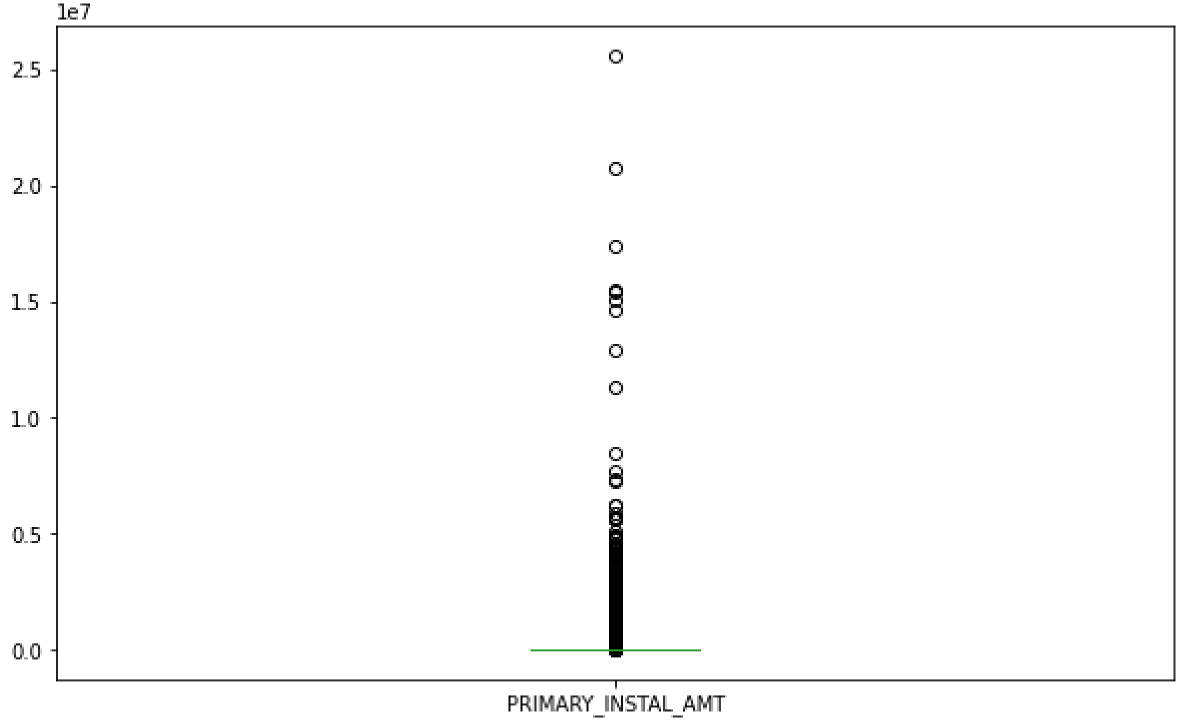
In [48]:
```
fig23,ax23=plt.subplots(1,1,figsize=(10,6));
loan_df.SEC_DISBURSED_AMOUNT.plot.box(ax=ax23);
```



In [49]:
```
loan_df.SEC_DISBURSED_AMOUNT.value_counts()
```

Out[49]:
```
0          229450
50000          59
100000         47
200000         36
40000          31
            ...
141467          1
252785          1
136000          1
39110           1
360499          1
Name: SEC_DISBURSED_AMOUNT, Length: 2553, dtype: int64
```

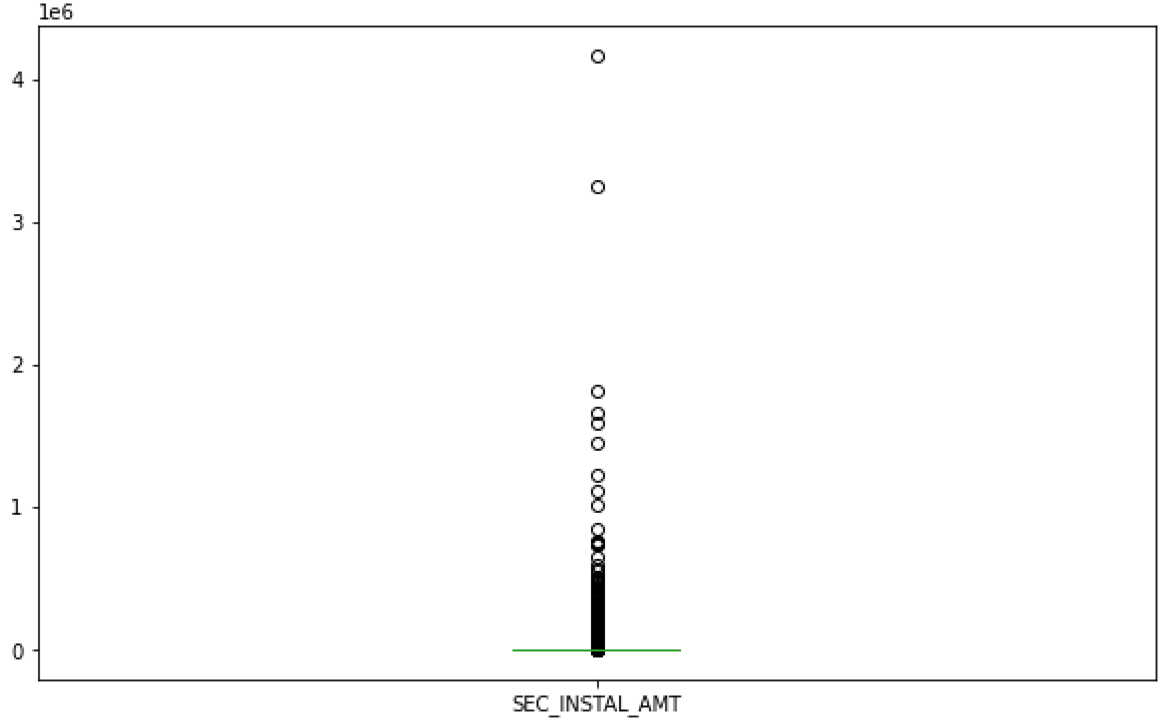
In [50]:
```
fig24,ax24=plt.subplots(1,1,figsize=(10,6));
loan_df.PRIMARY_INSTAL_AMT.plot.box(ax=ax24);
```



In [51]:
```
loan_df.PRIMARY_INSTAL_AMT.value_counts()
```

Out[51]:
```
0         159517
1620         292
1500         156
1600         144
2000         141
           ...
102786         1
33778          1
26603          1
14425          1
293886         1
Name: PRIMARY_INSTAL_AMT, Length: 28067, dtype: int64
```

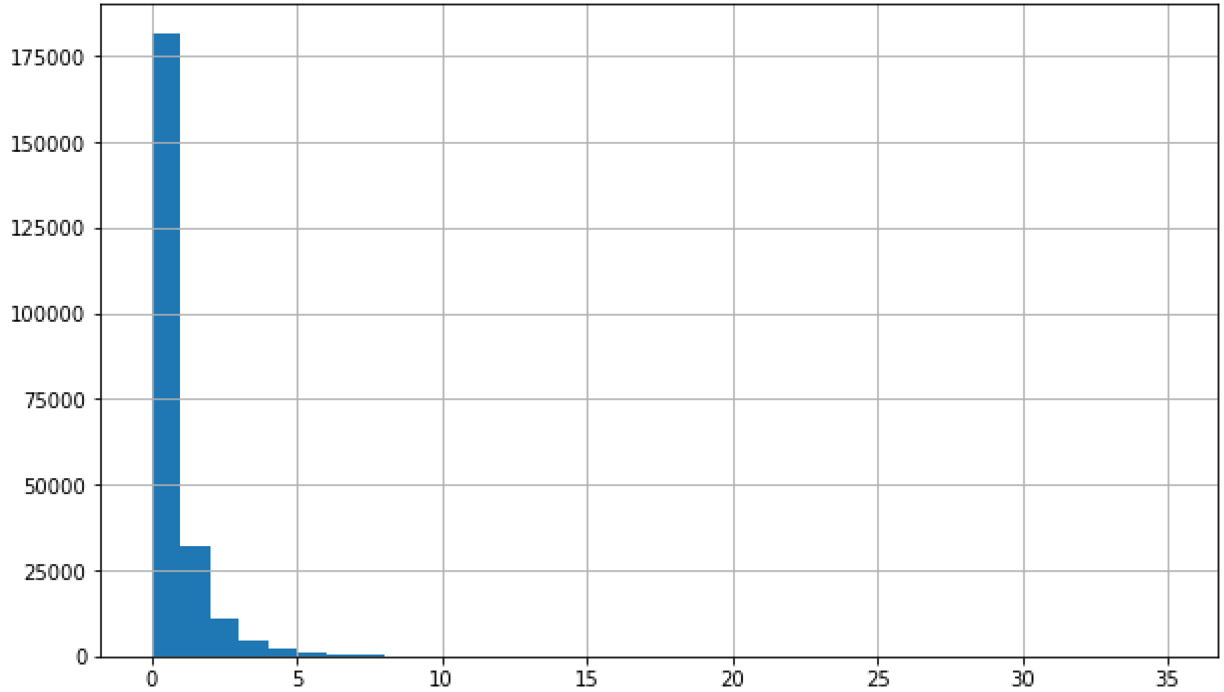
In [52]:
```
fig25,ax25=plt.subplots(1,1,figsize=(10,6));
loan_df.SEC_INSTAL_AMT.plot.box(ax=ax25);
```



In [53]:
```
loan_df.SEC_INSTAL_AMT.value_counts()
```

Out[53]:
```
0        230937
2100          7
5000          6
1065          6
1100          6
          ...  
7595          1
6971          1
1529          1
14260         1
49956         1
Name: SEC_INSTAL_AMT, Length: 1918, dtype: int64
```

In [54]:
```
fig26,ax26=plt.subplots(1,1,figsize=(10,6));
loan_df.NEW_ACCTS_IN_LAST_SIX_MONTHS.hist(ax=ax26,bins=35);
```



In [55]:

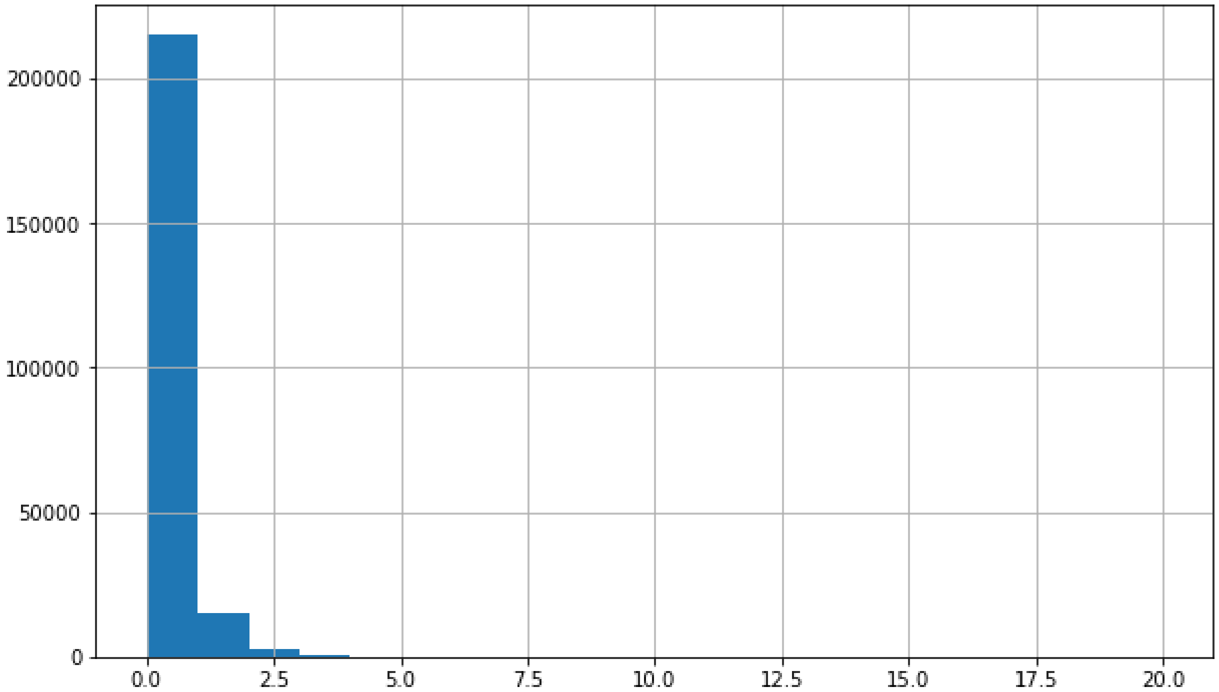
```
loan_df.NEW_ACCTS_IN_LAST_SIX_MONTHS.value_counts()
```

Out[55]:

```
0     181494
1      32099
2      11015
3       4458
4       1957
5        964
6        480
7        302
8        147
9         79
10        55
11        31
12        20
13        15
14        11
16         6
17         6
20         3
```

```
18         2
15         2
19         2
23         2
22         1
21         1
28         1
35         1
Name: NEW_ACCTS_IN_LAST_SIX_MONTHS, dtype: int64
```

In [56]:
```
fig27,ax27=plt.subplots(1,1,figsize=(10,6));
loan_df.DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS.hist(ax=ax27,bins=20);
```



In [57]:
```
loan_df.DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS.value_counts()
```

Out[57]:
```
0     214959
1      14941
2       2470
3        537
```

```
4         138
5          58
6          20
7          13
8           7
11          3
12          3
10          2
9           2
20          1
Name: DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS, dtype: int64
```

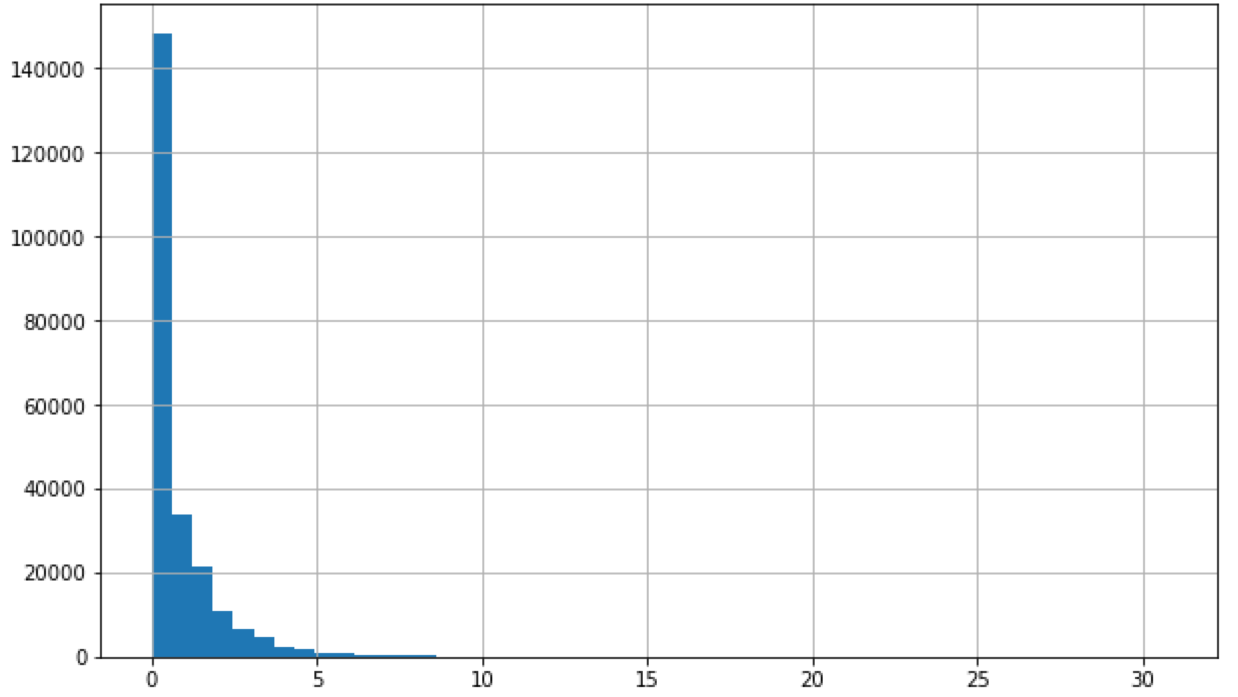
In [ ]:

In [58]:

```
loan_df.AVERAGE_ACCT_AGE=[float(i.split()[0][:-3])+float(i.split()[1][:-3])/12 for i in loan_df.AVERAGE_ACCT_AGE];
```

In [59]:

```
fig28,ax28=plt.subplots(1,1,figsize=(10,6));
loan_df.AVERAGE_ACCT_AGE.hist(ax=ax28,bins=50);
```

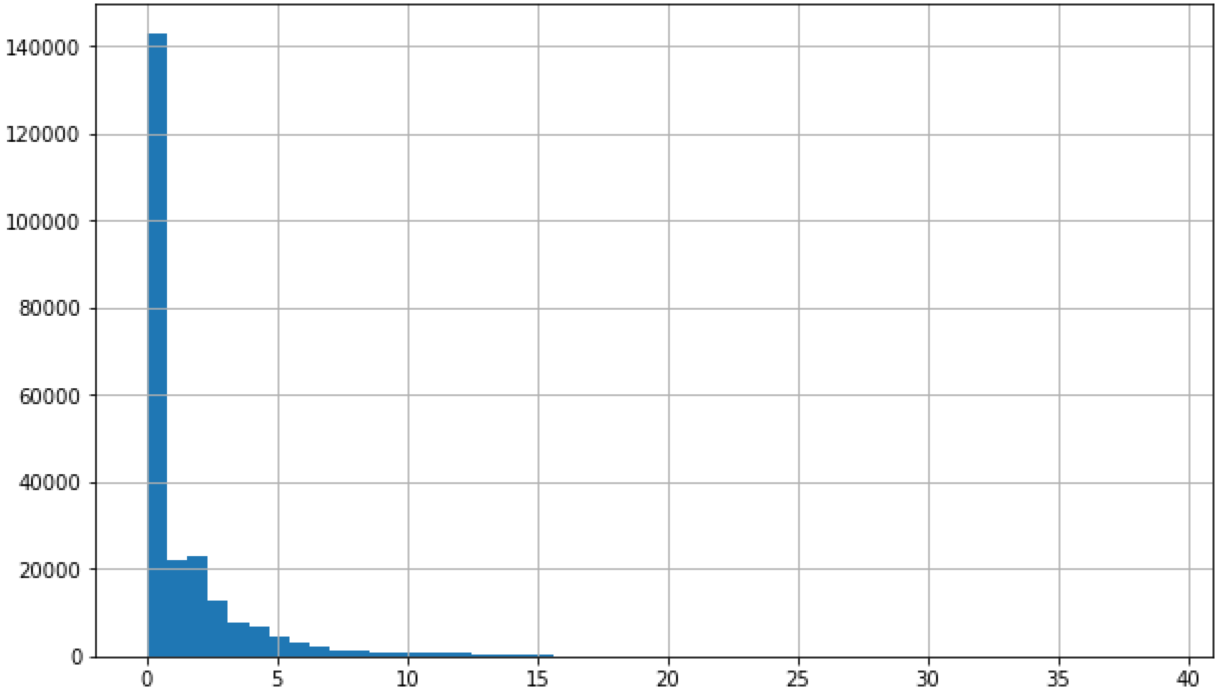


In [60]:

```
loan_df.CREDIT_HISTORY_LENGTH=[float(i.split()[0][:-3])+float(i.split()[1][:-3])/12 for i in loan_df.CREDIT_HISTORY_LENGTH];
```

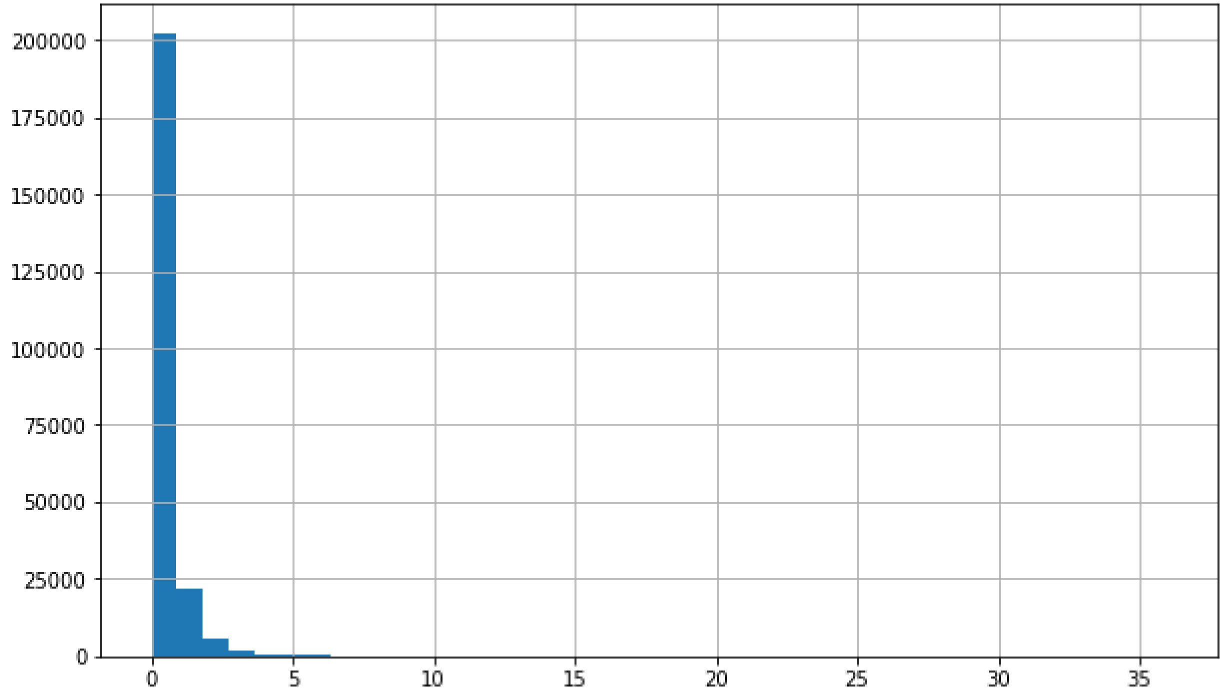
In [61]:

```
fig29,ax29=plt.subplots(1,1,figsize=(10,6));
loan_df.CREDIT_HISTORY_LENGTH.hist(ax=ax29,bins=50);
```



In [62]:

```
fig30,ax30=plt.subplots(1,1,figsize=(10,6));
loan_df.NO_OF_INQUIRIES.hist(ax=ax30,bins=40);
```



In [63]:

```
loan_df.NO_OF_INQUIRIES.value_counts()
```

Out[63]:

```
0     201961
1      22285
2       5409
3       1767
4        760
5        343
6        239
7        135
8        105
9         44
10        34
11        15
12        14
14         8
15         7
19         6
13         6
17         4
```

```
18          4
16          3
28          1
20          1
23          1
36          1
22          1
Name: NO_OF_INQUIRIES, dtype: int64
```

5. Explain how is the target variable distributed overall

In [64]:

```
loan_df.loan_default.value_counts()
```

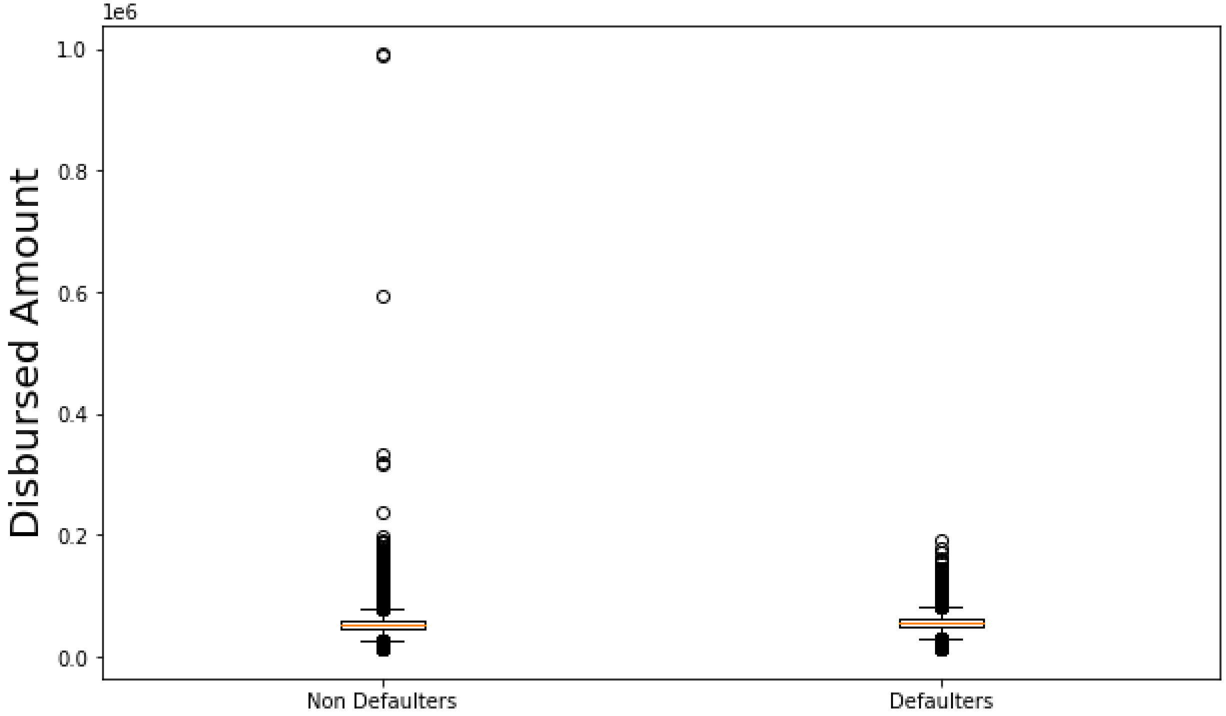
Out[64]:

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

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

In [65]:

```
fig31,ax31=plt.subplots(1,1,figsize=(10,6));
ax31.boxplot([loan_df.disbursed_amount[loan_df.loan_default==0],loan_df.disbursed_amount[loan_df.loan_default==1]]);
ax31.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax31.set_ylabel('Disbursed Amount',size=20);
```

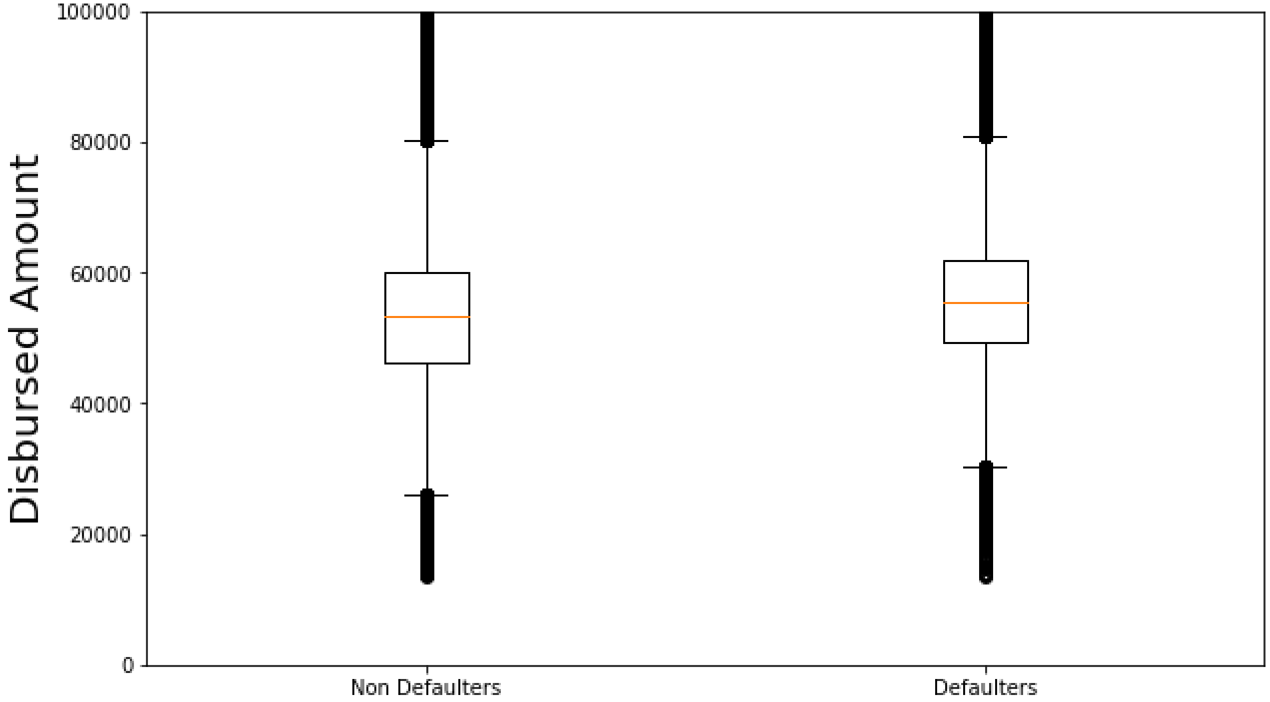


In [66]:

```
fig31,ax31=plt.subplots(1,1,figsize=(10,6));
ax31.boxplot([loan_df.disbursed_amount[loan_df.loan_default==0],loan_df.disbursed_amount[loan_df.loan_default==1]]);
ax31.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax31.set_ylabel('Disbursed Amount',size=20);
ax31.set_ylim([0,100000])
```

Out[66]:

```
(0.0, 100000.0)
```

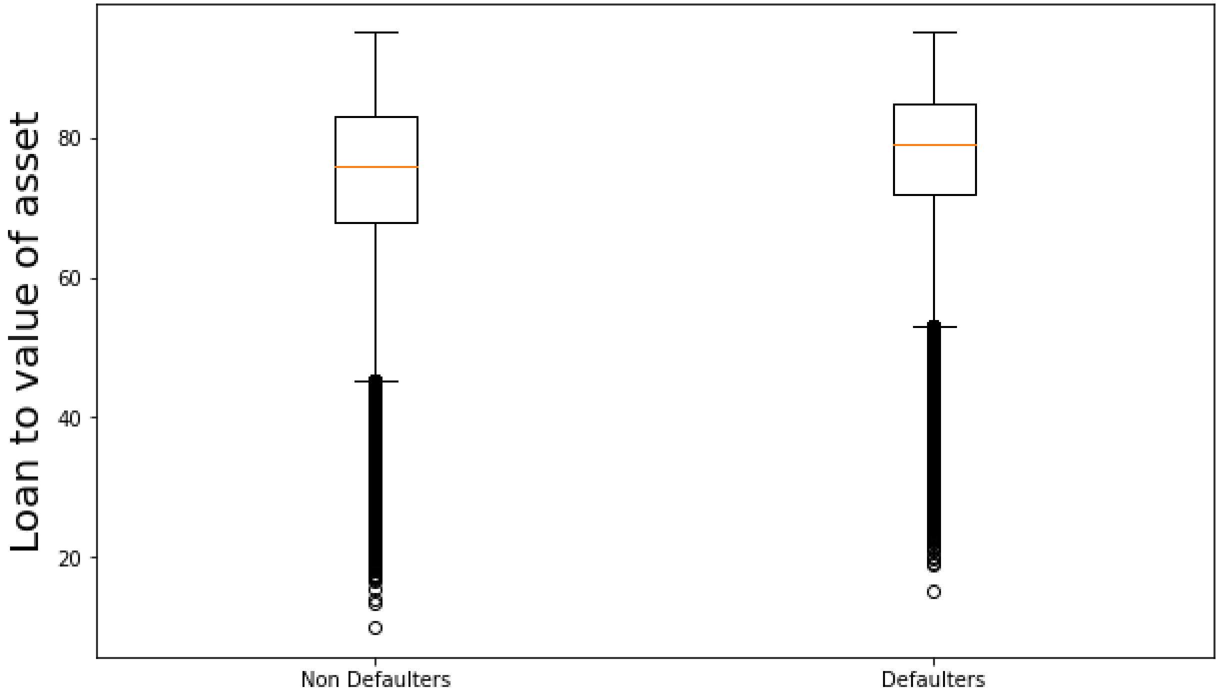


In [67]:
```
spearmanr(loan_df.disbursed_amount,loan_df.loan_default)
```

Out[67]:
```
SpearmanrResult(correlation=0.09288435655814552, pvalue=0.0)
```

In [ ]:

In [68]:
```
fig32,ax32=plt.subplots(1,1,figsize=(10,6));
ax32.boxplot([loan_df.ltv[loan_df.loan_default==0],loan_df.ltv[loan_df.loan_default==1]]);
ax32.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax32.set_ylabel('Loan to value of asset',size=20);
```



In [69]:
```
spearmanr(loan_df.ltv,loan_df.loan_default)
```

Out[69]: SpearmanrResult(correlation=0.09908847004863419, pvalue=0.0)

In [70]:
```
branch_count=pd.merge(loan_df[loan_df.loan_default==0].branch_id.value_counts(),loan_df[loan_df.loan_default==1].branch_id.value_c
```

In [71]:
```
branch_count.rename(columns={'branch_id_x':'num_non_defaulters','branch_id_y':'num_defaulters','key_0':'branch_number'},inplace=Tr
branch_count.set_index(branch_count.branch_number, inplace=True)
branch_count.drop(columns=['branch_number'], inplace=True)
branch_count.sort_index(inplace=True)
```

In [72]:
```
branch_count['defaulter_ratio']=branch_count.num_defaulters/(branch_count.num_non_defaulters+branch_count.num_defaulters);
```

In [73]:
```
len(loan_df[loan_df.loan_default==1])/len(loan_df)
```

Out[73]:
```
0.2170711203753742
```

In [74]:
```
branch_count.sort_values(by=['defaulter_ratio'],ascending=False,inplace=True)
```

In [75]:
```
branch_count
```

Out[75]:

branch_number	num_non_defaulters	num_defaulters	defaulter_ratio
158	50	19	0.275362
258	297	98	0.248101
217	138	45	0.245902
35	500	158	0.240122
69	595	187	0.239130
...	...	...	...
73	1158	279	0.194154
254	1148	269	0.189838
17	989	227	0.186678
66	261	52	0.166134
259	267	53	0.165625

82 rows × 3 columns

In [76]:
```
loan_df.manufacturer_id.value_counts()
```

Out[76]:
```
86     109534
45      56626
51      27204
48      16710
49      10220
120      9658
```

```
67       2405
145       778
153        12
152         6
156         1
Name: manufacturer_id, dtype: int64
```

In [77]:
```
loan_df[loan_df.loan_default==0].manufacturer_id.value_counts()
```

Out[77]:
```
86     87124
45     43687
51     21547
48     12156
49      7984
120     7526
67      1882
145      622
153        8
152        6
156        1
Name: manufacturer_id, dtype: int64
```

In [78]:
```
loan_df[loan_df.loan_default==1].manufacturer_id.value_counts()
```

Out[78]:
```
86     22410
45     12939
51      5657
48      4554
49      2236
120     2132
67       523
145      156
153        4
Name: manufacturer_id, dtype: int64
```

In [79]:
```
loan_df.manufacturer_id.value_counts().index
```

Out[79]:
```
Int64Index([86, 45, 51, 48, 49, 120, 67, 145, 153, 152, 156], dtype='int64')
```

In [80]:
```
manufacturer_cnt=pd.merge(loan_df[loan_df.loan_default==0].manufacturer_id.value_counts(),loan_df[loan_df.loan_default==1].manufac
```

In [81]:
```
manufacturer_cnt.fillna(0,inplace=True);
manufacturer_cnt.rename(columns={'manufacturer_id_x':'num_non_defaulters','manufacturer_id_y':'num_defaulters'},inplace=True)
manufacturer_cnt.sort_index(inplace=True)
```

In [82]:
```
manufacturer_cnt['defaulter_ratio']=manufacturer_cnt.num_defaulters/(manufacturer_cnt.num_non_defaulters+manufacturer_cnt.num_defa
manufacturer_cnt.sort_values(by=['defaulter_ratio'],ascending=False,inplace=True)
```

In [83]:
```
manufacturer_cnt
```

Out[83]:

	num_non_defaulters	num_defaulters	defaulter_ratio
153	8	4.0	0.333333
48	12156	4554.0	0.272531
45	43687	12939.0	0.228499
120	7526	2132.0	0.220750
49	7984	2236.0	0.218787
67	1882	523.0	0.217464
51	21547	5657.0	0.207947
86	87124	22410.0	0.204594
145	622	156.0	0.200514
152	6	0.0	0.000000
156	1	0.0	0.000000

In [84]:
```
supplier_cnt=pd.merge(loan_df[loan_df.loan_default==0].supplier_id.value_counts(),loan_df[loan_df.loan_default==1].supplier_id.val
```

In [85]:
```
supplier_cnt.fillna(0,inplace=True);
supplier_cnt.rename(columns={'supplier_id_x':'num_non_defaulters','supplier_id_y':'num_defaulters'},inplace=True)
supplier_cnt.sort_index(inplace=True)
```

In [86]:
```
supplier_cnt['defaulter_ratio']=supplier_cnt.num_defaulters/(supplier_cnt.num_non_defaulters+supplier_cnt.num_defaulters);
supplier_cnt.sort_values(by=['defaulter_ratio'],ascending=False,inplace=True)
```

In [87]:
```
supplier_cnt.head(60)
```

Out[87]:

	num_non_defaulters	num_defaulters	defaulter_ratio
15045	0.0	1.0	1.000000
24109	0.0	3.0	1.000000
23685	0.0	2.0	1.000000
23741	0.0	1.0	1.000000
18513	0.0	1.0	1.000000
23802	0.0	1.0	1.000000
23932	0.0	1.0	1.000000
18102	0.0	1.0	1.000000
18099	0.0	2.0	1.000000
22474	0.0	1.0	1.000000
23635	0.0	1.0	1.000000
24222	0.0	1.0	1.000000
24252	0.0	1.0	1.000000
17865	0.0	1.0	1.000000
17228	0.0	1.0	1.000000
17183	0.0	1.0	1.000000
17129	0.0	1.0	1.000000
24443	0.0	1.0	1.000000
23661	0.0	2.0	1.000000
20315	0.0	1.0	1.000000

	num_non_defaulters	num_defaulters	defaulter_ratio
24552	0.0	1.0	1.000000
23088	0.0	1.0	1.000000
22630	0.0	1.0	1.000000
22751	0.0	1.0	1.000000
22840	0.0	2.0	1.000000
22845	0.0	1.0	1.000000
22859	0.0	1.0	1.000000
21981	0.0	1.0	1.000000
21847	0.0	1.0	1.000000
21511	0.0	2.0	1.000000
23541	0.0	1.0	1.000000
23111	0.0	2.0	1.000000
20943	0.0	1.0	1.000000
23171	0.0	2.0	1.000000
23188	0.0	1.0	1.000000
23189	0.0	1.0	1.000000
20931	0.0	1.0	1.000000
20763	0.0	2.0	1.000000
16788	0.0	2.0	1.000000
22552	0.0	1.0	1.000000
24742	0.0	1.0	1.000000
24790	0.0	1.0	1.000000
24599	0.0	3.0	1.000000
24724	0.0	1.0	1.000000

	num_non_defaulters	num_defaulters	defaulter_ratio
24700	0.0	1.0	1.000000
24679	0.0	1.0	1.000000
24616	0.0	1.0	1.000000
24789	0.0	1.0	1.000000
24793	0.0	1.0	1.000000
24715	0.0	1.0	1.000000
24713	0.0	2.0	1.000000
15186	0.0	1.0	1.000000
24555	0.0	1.0	1.000000
24598	0.0	1.0	1.000000
24294	1.0	4.0	0.800000
24579	2.0	7.0	0.777778
23356	5.0	16.0	0.761905
24699	1.0	3.0	0.750000
24534	1.0	3.0	0.750000
24200	9.0	24.0	0.727273

In [88]:

```
supplier_cnt.tail(60)
```

Out[88]:

	num_non_defaulters	num_defaulters	defaulter_ratio
22985	2.0	0.0	0.0
21314	2.0	0.0	0.0
23005	8.0	0.0	0.0
23406	1.0	0.0	0.0

	num_non_defaulters	num_defaulters	defaulter_ratio
21149	1.0	0.0	0.0
18123	13.0	0.0	0.0
24581	1.0	0.0	0.0
23404	18.0	0.0	0.0
23038	1.0	0.0	0.0
23037	2.0	0.0	0.0
17323	1.0	0.0	0.0
23028	3.0	0.0	0.0
17790	1.0	0.0	0.0
23025	4.0	0.0	0.0
24570	8.0	0.0	0.0
23022	2.0	0.0	0.0
24568	2.0	0.0	0.0
24206	3.0	0.0	0.0
17394	15.0	0.0	0.0
24565	1.0	0.0	0.0
24564	4.0	0.0	0.0
22980	1.0	0.0	0.0
24530	4.0	0.0	0.0
24529	9.0	0.0	0.0
21771	1.0	0.0	0.0
22942	1.0	0.0	0.0
14622	40.0	0.0	0.0
24492	4.0	0.0	0.0

	num_non_defaulters	num_defaulters	defaulter_ratio
24041	7.0	0.0	0.0
21843	1.0	0.0	0.0
23885	5.0	0.0	0.0
18045	2.0	0.0	0.0
14331	1.0	0.0	0.0
24248	1.0	0.0	0.0
22938	2.0	0.0	0.0
17451	2.0	0.0	0.0
24479	5.0	0.0	0.0
15775	6.0	0.0	0.0
24476	20.0	0.0	0.0
24474	2.0	0.0	0.0
24498	1.0	0.0	0.0
15779	2.0	0.0	0.0
24525	5.0	0.0	0.0
23425	1.0	0.0	0.0
24229	4.0	0.0	0.0
24522	3.0	0.0	0.0
21422	1.0	0.0	0.0
23422	5.0	0.0	0.0
24518	3.0	0.0	0.0
22956	1.0	0.0	0.0
24234	1.0	0.0	0.0
21597	3.0	0.0	0.0

	num_non_defaulters	num_defaulters	defaulter_ratio
24235	2.0	0.0	0.0
23825	8.0	0.0	0.0
21683	1.0	0.0	0.0
21702	1.0	0.0	0.0
15383	2.0	0.0	0.0
22954	1.0	0.0	0.0
21703	12.0	0.0	0.0
23336	22.0	0.0	0.0

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

In [89]:
```
loan_df[loan_df.Employment_Type.isna()].Employee_code_ID.value_counts()
```

Out[89]:
```
908     62
1660    43
194     42
1192    40
140     39
        ..
327      1
2686     1
2629     1
1082     1
199      1
Name: Employee_code_ID, Length: 1379, dtype: int64
```

In [90]:
```
loan_df.Employee_code_ID.value_counts()[908]
```

Out[90]:
```
341
```

In [91]:
```
loan_df[loan_df.Employment_Type=='Salaried'].Employee_code_ID.value_counts()[908]
```
Out[91]: 278

In [92]:
```
loan_df[loan_df.Employment_Type=='Self employed'].Employee_code_ID.value_counts()[908]
```
Out[92]: 1

In [93]:
```
loan_df.Employee_code_ID.value_counts()[194]
```
Out[93]: 191

In [94]:
```
loan_df[loan_df.Employment_Type=='Salaried'].Employee_code_ID.value_counts()[194]
```
Out[94]: 147

In [95]:
```
loan_df[loan_df.Employment_Type=='Self employed'].Employee_code_ID.value_counts()[194]
```
Out[95]: 2

In [96]:
```
loan_df.Employee_code_ID.value_counts()[1192]
```
Out[96]: 145

In [97]:
```
loan_df[loan_df.Employment_Type=='Salaried'].Employee_code_ID.value_counts()[1192]
```
Out[97]: 76

In [98]:
```
loan_df[loan_df.Employment_Type=='Self employed'].Employee_code_ID.value_counts()[1192]
```
Out[98]: 29

In [99]:

```
loan_df.Employee_code_ID.value_counts()[140]
```

Out[99]:

```
185
```

In [100…

```
loan_df[loan_df.Employment_Type=='Salaried'].Employee_code_ID.value_counts()[140]
```

Out[100…

```
122
```

In [101…

```
loan_df[loan_df.Employment_Type=='Self employed'].Employee_code_ID.value_counts()[140]
```

Out[101…

```
24
```

Mode value of employee code id is a good way to assign Emplyee Type for na values

In [102…

```
emp_mapping=dict();
for code_id in loan_df[loan_df.Employment_Type.isna()].Employee_code_ID.unique():
    emp_mapping[code_id]=str(loan_df[loan_df.Employee_code_ID==code_id]['Employment_Type'].mode()).split('\n')[0].split('0')[-1].s
```

In [103…

```
loan_df.Employment_Type = loan_df.Employment_Type.fillna(loan_df.Employee_code_ID.map(emp_mapping))
```

In [104…

```
loan_df.Employment_Type.value_counts()
```

Out[104…

```
Self employed                                      131849
Salaried                                           101304
Series([], Name: Employment_Type, dtype: object)        1
Name: Employment_Type, dtype: int64
```

In [105…

```
loan_df.Employment_Type.mode()[0]
```

Out[105…

```
'Self employed'
```

In [106…

```
loan_df.Employment_Type.replace(loan_df.loc[loan_df[(loan_df.Employment_Type!='Self employed') & (loan_df.Employment_Type!='Salari
```

In [107…

```
loan_df.Employment_Type.value_counts()
```

Out[107…

```
Self employed    131850
Salaried         101304
Name: Employment_Type, dtype: int64
```

In [108…

```
loan_df[loan_df.loan_default==0].Employment_Type.value_counts()
```

Out[108…

```
Self employed    101887
Salaried          80656
Name: Employment_Type, dtype: int64
```

In [109…

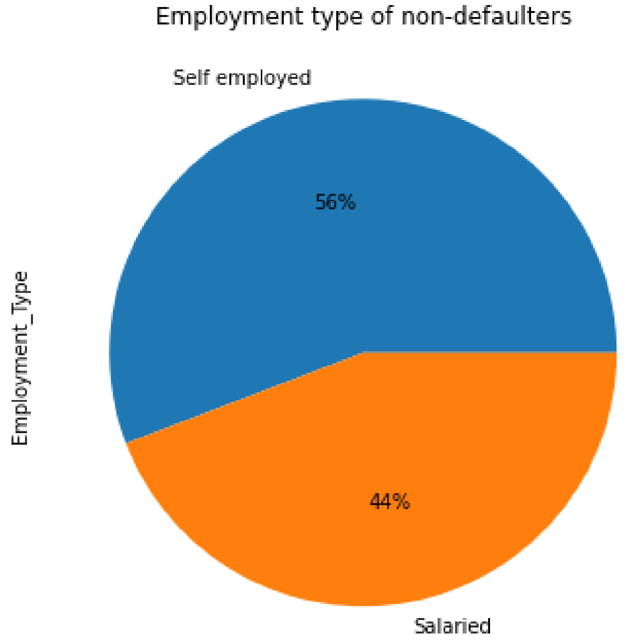
```
loan_df[loan_df.loan_default==1].Employment_Type.value_counts()
```

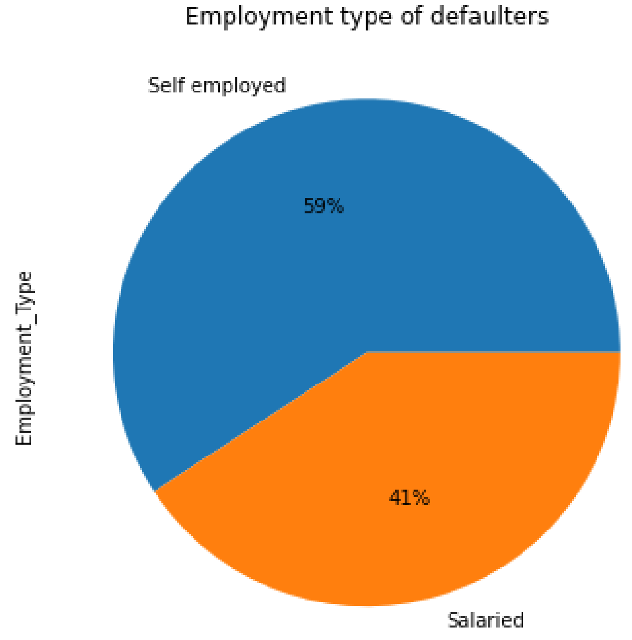
Out[109…

```
Self employed    29963
Salaried         20648
Name: Employment_Type, dtype: int64
```

In [110…

```
fig33,ax33=plt.subplots(1,2,figsize=(20,6));
loan_df[loan_df.loan_default==0].Employment_Type.value_counts().plot(kind='pie',autopct='%1.0f%%',ax=ax33[0]);
loan_df[loan_df.loan_default==1].Employment_Type.value_counts().plot(kind='pie',autopct='%1.0f%%',ax=ax33[1]);
ax33[0].set_title('Employment type of non-defaulters');
ax33[1].set_title('Employment type of defaulters');
```

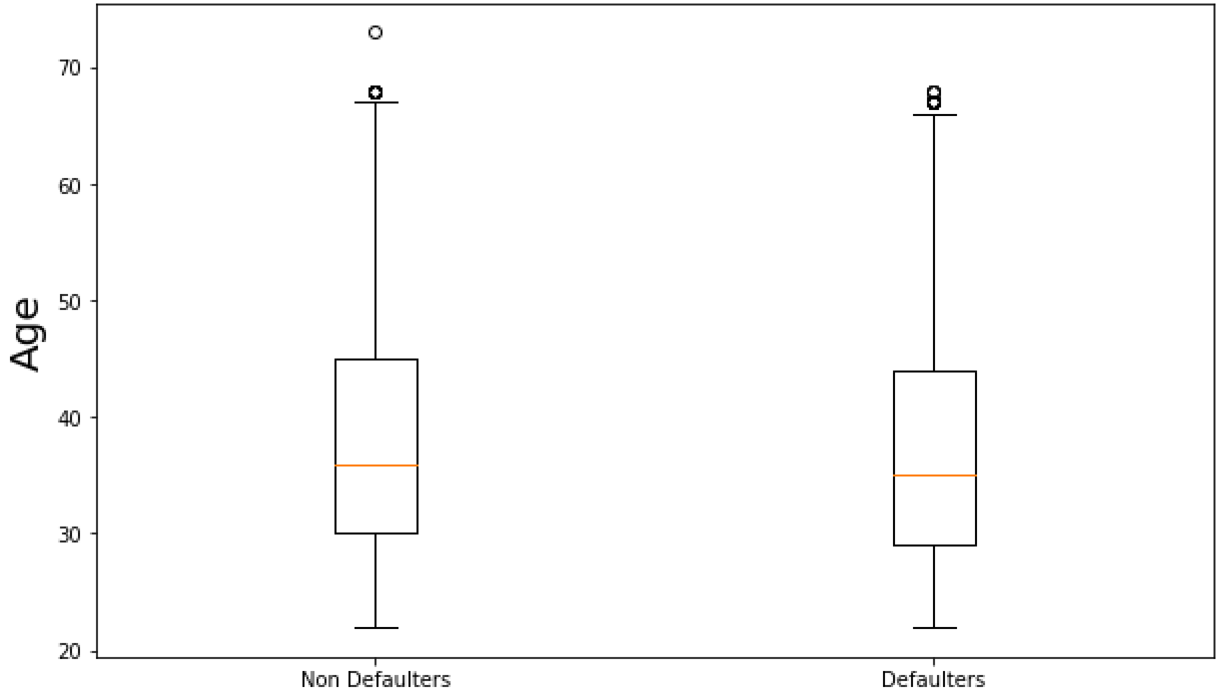




8. Has age got anything to do with defaulting? What is the distribution of age w.r.t. to the defaulters and non-defaulters?

In [111…

```
fig34,ax34=plt.subplots(1,1,figsize=(10,6));
ax34.boxplot([loan_df.age[loan_df.loan_default==0],loan_df.age[loan_df.loan_default==1]]);
ax34.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax34.set_ylabel('Age',size=20);
```



9. What type of ID was presented by most of the customers for proof?

In [112…

```
print("Percentage of customers who gave their mobile number is = " ,len(loan_df[loan_df.MobileNo_Avl_Flag==1])/len(loan_df)*100,"%
```

```
Percentage of customers who gave their mobile number is =  100.0 %
```

In [113…

```
print("Percentage of customers who gave their Aadhar card is = " ,len(loan_df[loan_df.Aadhar_flag==1])/len(loan_df)*100,"%")
```

```
Percentage of customers who gave their Aadhar card is =  84.03201317584086 %
```

In [114…

```
print("Percentage of customers who gave their PAN card is = " ,len(loan_df[loan_df.PAN_flag==1])/len(loan_df)*100,'%')
```

```
Percentage of customers who gave their PAN card is =  7.557665748818378 %
```

In [115…

```
print("Percentage of customers who gave their Voter ID card is = " ,len(loan_df[loan_df.VoterID_flag==1])/len(loan_df)*100,'%')
```

```
Percentage of customers who gave their Voter ID card is =  14.494282748741174 %
```

In [116…
```
print("Percentage of customers who gave their Driving license card is = " ,len(loan_df[loan_df.Driving_flag==1])/len(loan_df)*100,
```

```
Percentage of customers who gave their Driving license card is =  2.3242148965919522 %
```

In [117…
```
print("Percentage of customers who gave their Passport card is = " ,len(loan_df[loan_df.Passport_flag==1])/len(loan_df)*100,'%')
```

```
Percentage of customers who gave their Passport card is =  0.21273493056091683 %
```

While all the customers presented their mobile number, most of them presented their Aadhar card for proof (84.03%)

Observations based on Week 1 analysis:

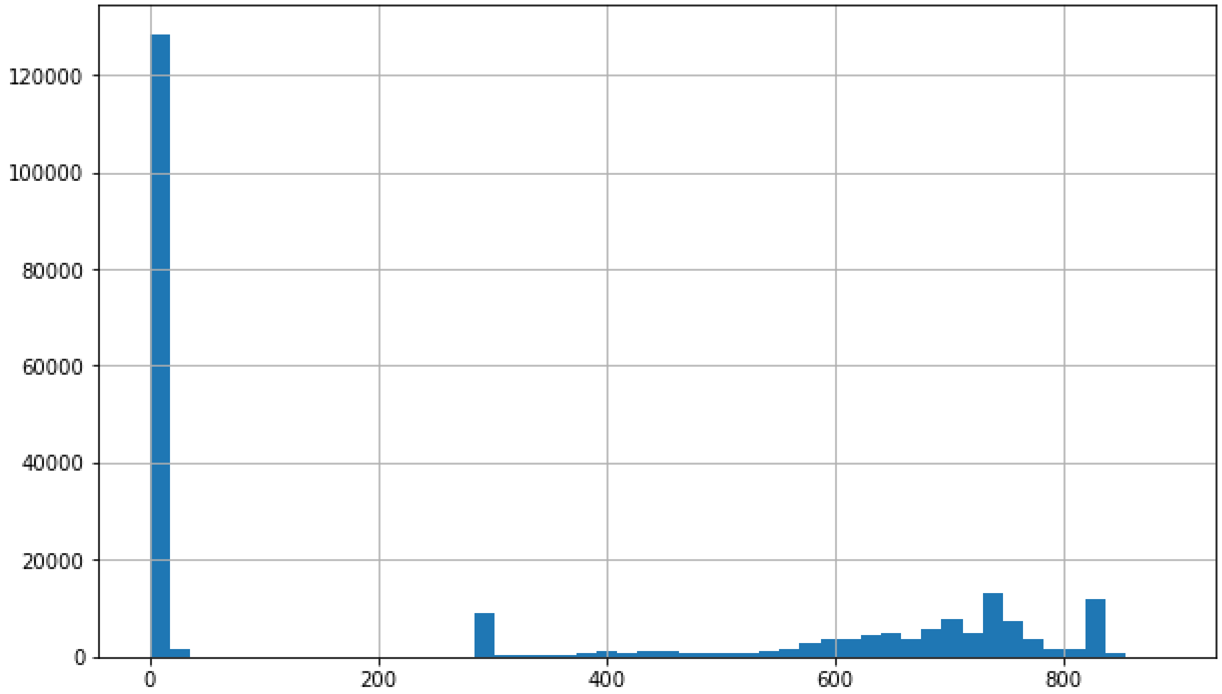
1. There are a total of 233154 customer observations. The data has a total of 40 features and 1 target value indicating if there is loan default or not.
2. 'Employee_Type' parameter has 7661 null values. Remaining employess are classfied as "Self-Employed" and "Salaried". A good strategy to fill missing 'Emplyee_Type' values is to check the 'Employee_code_ID' and fill missing values accordingly as the mode of 'Employee_Type' for corresponding 'Employee_code_ID'. For example, 62 people with missing Employee_Type have Employee_Code_ID as 908. Among remaining customers, 278 salaried individuals have code ID 908 and only 1 self-employed customer has code ID 908. This suggests that majority of Salaries employees have employee Code id 908. Therefore, it would be safe to mention 'Salaried' as the 'Employee_Type' for missing values with 'Employee_code_ID' of 908.
3. Most customers have Disbursed amount and asset cost of 100,000 or less while these observations for others is much greater.
4. Most borrowers are young while as the age increases the number of borrowers decreases.
5. Most customers have very few primary or secondary accounts.
6. Most customers have average account age and credit history length of less than 5 years.
7. While 182543 are not defaulters, remaining 50611 are defaulters and need to be identified using appropriate model.
8. Customers who take survice from certain vehicle manufacturers and suppliers have higher tendency to default compared to other manfucturers and suppliers
9. Customers with higher loan to value of asset ratio have higher tendency to default.
10. Customers with higher disbursed amount have higher tendency to default.
11. Though age distribution of defaulters and non-defaulters is similar, the boxplots suggest that the age distrbution of defaulters tends to be marginally lower compared to no-defaulters.
12. While all the customers presented their mobile number, most of them presented their Aadhar card for proof (84.03%) and some others (14.49%) presented their Voter ID. However, very few customers present their Driving License, PAN card or passport.

Week 2: Performing EDA and Modeling

1. Study the credit bureau score distribution. Compare the distribution for defaulters vs. non-defaulters. Explore in detail.

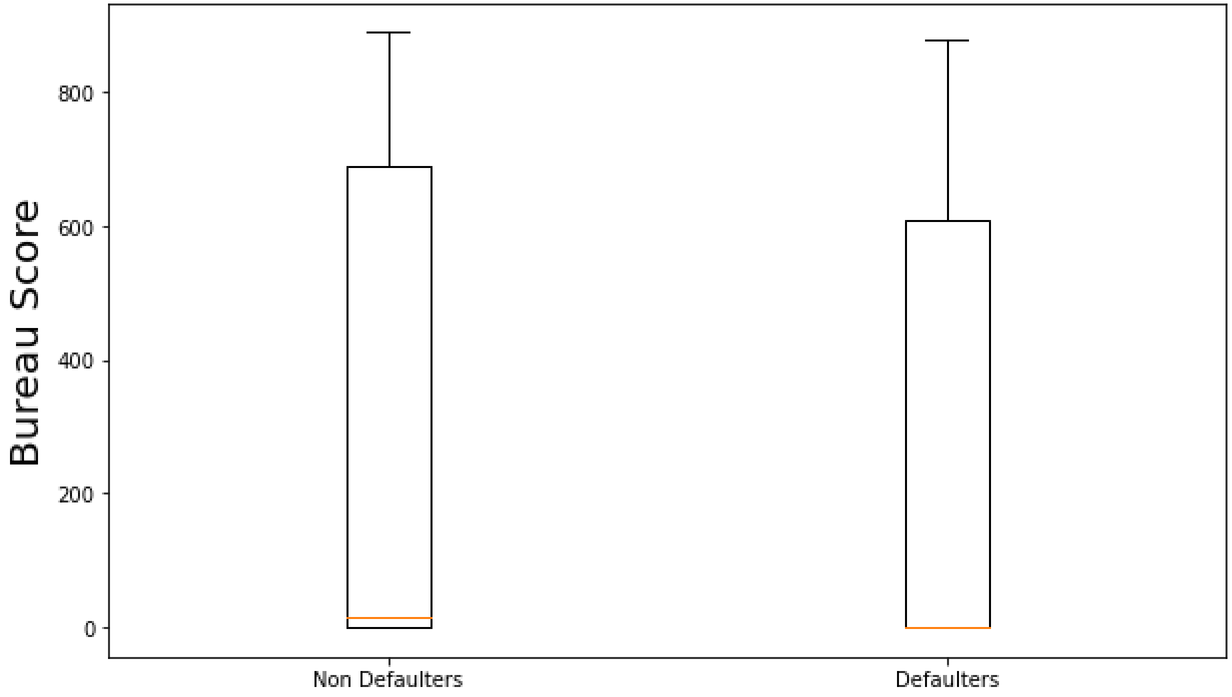
In [118…

```
fig35,ax35=plt.subplots(1,1,figsize=(10,6));
loan_df.PERFORM_CNS_SCORE.hist(ax=ax35,bins=50);
```



In [119…

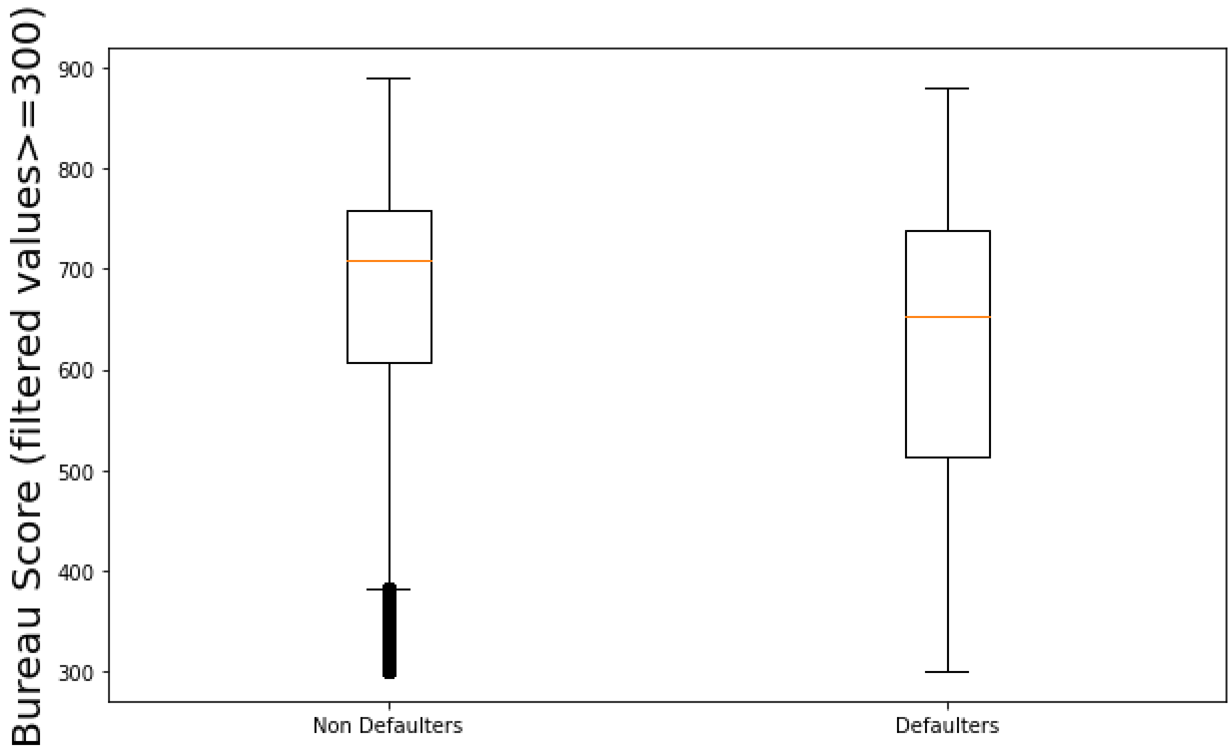
```
fig36,ax36=plt.subplots(1,1,figsize=(10,6));
ax36.boxplot([loan_df.PERFORM_CNS_SCORE[loan_df.loan_default==0],loan_df.PERFORM_CNS_SCORE[loan_df.loan_default==1]]);
ax36.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax36.set_ylabel('Bureau Score',size=20);
```



In [ ]:

In [120…

```
fig36,ax36=plt.subplots(1,1,figsize=(10,6));
ax36.boxplot([loan_df.PERFORM_CNS_SCORE[(loan_df.loan_default==0)&(loan_df.PERFORM_CNS_SCORE>=300)],loan_df.PERFORM_CNS_SCORE[(loa
ax36.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax36.set_ylabel('Bureau Score (filtered values>=300)',size=20);
```



In [121…

```
spearmanr(loan_df.PERFORM_CNS_SCORE[(loan_df.PERFORM_CNS_SCORE>=300)],loan_df.loan_default[(loan_df.PERFORM_CNS_SCORE>=300)])
```

Out[121…

```
SpearmanrResult(correlation=-0.12443220768045092, pvalue=0.0)
```

In [122…

```
loan_df.PERFORM_CNS_SCORE_DESCRIPTION.value_counts()
```

Out[122…

```
No Bureau History Available                              116950
C-Very Low Risk                                           16045
A-Very Low Risk                                           14124
D-Very Low Risk                                           11358
B-Very Low Risk                                            9201
M-Very High Risk                                           8776
F-Low Risk                                                 8485
K-High Risk                                                8277
H-Medium Risk                                              6855
E-Low Risk                                                 5821
I-Medium Risk                                              5557
```

```
G-Low Risk                                              3988
Not Scored: Sufficient History Not Available            3765
J-High Risk                                             3748
Not Scored: Not Enough Info available on the customer   3672
Not Scored: No Activity seen on the customer (Inactive) 2885
Not Scored: No Updates available in last 36 months      1534
L-Very High Risk                                        1134
Not Scored: Only a Guarantor                             976
Not Scored: More than 50 active Accounts found             3
Name: PERFORM_CNS_SCORE_DESCRIPTION, dtype: int64
```

In [123…

```
non_default_bureau_rating=loan_df.PERFORM_CNS_SCORE_DESCRIPTION[(loan_df.loan_default==0)].value_counts()
```

In [124…

```
default_bureau_rating=loan_df.PERFORM_CNS_SCORE_DESCRIPTION[(loan_df.loan_default==1)].value_counts()
```

In [125…

```
bureau_rating_dist=pd.merge(non_default_bureau_rating,default_bureau_rating ,how='left',left_index=True,right_index=True);
```

In [126…

```
bureau_rating_dist
```

Out[126…

	PERFORM_CNS_SCORE_DESCRIPTION_x	PERFORM_CNS_SCORE_DESCRIPTION_y
No Bureau History Available	89898	27052.0
C-Very Low Risk	13275	2770.0
A-Very Low Risk	11783	2341.0
D-Very Low Risk	9659	1699.0
B-Very Low Risk	7993	1208.0
F-Low Risk	6905	1580.0
M-Very High Risk	6103	2673.0
K-High Risk	5975	2302.0
H-Medium Risk	5197	1658.0
E-Low Risk	4821	1000.0

	PERFORM_CNS_SCORE_DESCRIPTION_x	PERFORM_CNS_SCORE_DESCRIPTION_y
I-Medium Risk	4042	1515.0
G-Low Risk	3202	786.0
Not Scored: Not Enough Info available on the customer	2902	770.0
Not Scored: Sufficient History Not Available	2802	963.0
J-High Risk	2802	946.0
Not Scored: No Activity seen on the customer (Inactive)	2355	530.0
Not Scored: No Updates available in last 36 months	1242	292.0
L-Very High Risk	816	318.0
Not Scored: Only a Guarantor	768	208.0
Not Scored: More than 50 active Accounts found	3	NaN

In [127…

```
bureau_rating_dist.fillna(0,inplace=True);
bureau_rating_dist.rename(columns={'PERFORM_CNS_SCORE_DESCRIPTION_x':'num_non_defaulters','PERFORM_CNS_SCORE_DESCRIPTION_y':'num_d
bureau_rating_dist.sort_index(inplace=True)
```

In [128…

```
bureau_rating_dist['defaulter_ratio']=bureau_rating_dist.num_defaulters/(bureau_rating_dist.num_non_defaulters+bureau_rating_dist.
bureau_rating_dist.sort_values(by=['defaulter_ratio'],ascending=False,inplace=True)
```

In [129…

```
bureau_rating_dist
```

Out[129…

	num_non_defaulters	num_defaulters	defaulter_ratio
M-Very High Risk	6103	2673.0	0.304581
L-Very High Risk	816	318.0	0.280423
K-High Risk	5975	2302.0	0.278120
I-Medium Risk	4042	1515.0	0.272629
Not Scored: Sufficient History Not Available	2802	963.0	0.255777

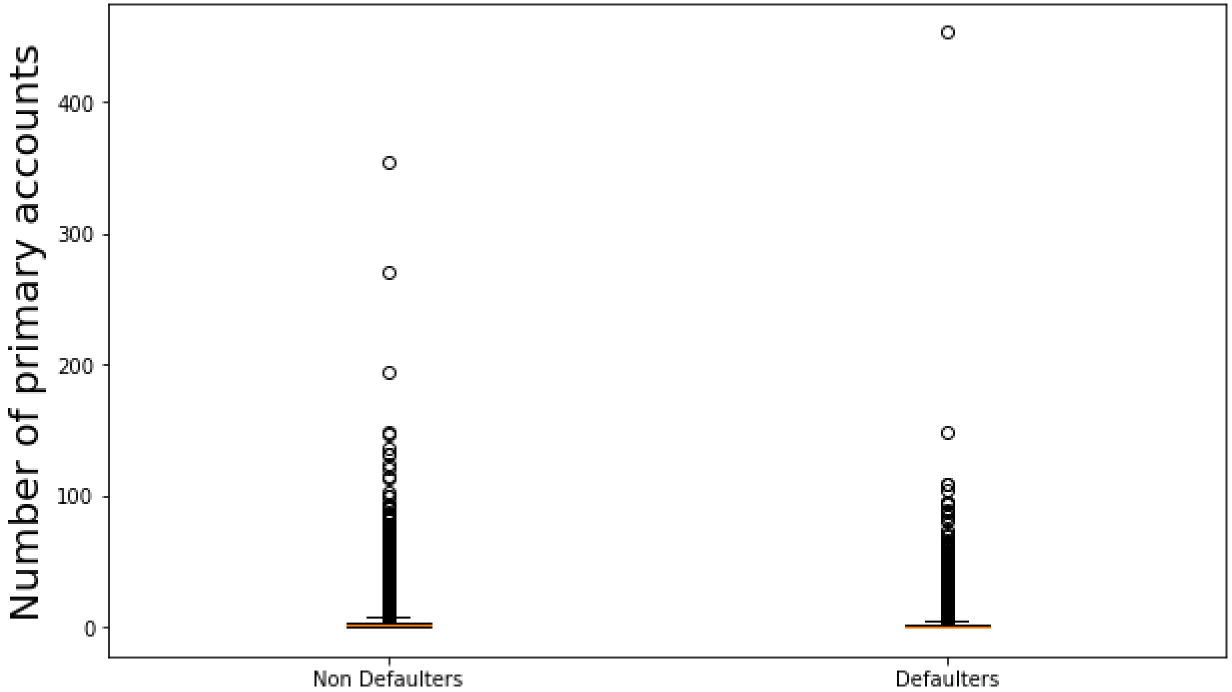
	num_non_defaulters	num_defaulters	defaulter_ratio
J-High Risk	2802	946.0	0.252401
H-Medium Risk	5197	1658.0	0.241867
No Bureau History Available	89898	27052.0	0.231313
Not Scored: Only a Guarantor	768	208.0	0.213115
Not Scored: Not Enough Info available on the customer	2902	770.0	0.209695
G-Low Risk	3202	786.0	0.197091
Not Scored: No Updates available in last 36 months	1242	292.0	0.190352
F-Low Risk	6905	1580.0	0.186211
Not Scored: No Activity seen on the customer (Inactive)	2355	530.0	0.183709
C-Very Low Risk	13275	2770.0	0.172639
E-Low Risk	4821	1000.0	0.171792
A-Very Low Risk	11783	2341.0	0.165746
D-Very Low Risk	9659	1699.0	0.149586
B-Very Low Risk	7993	1208.0	0.131290
Not Scored: More than 50 active Accounts found	3	0.0	0.000000

Boxplots show that non-defaulter have higher average bureau score compared to defaulters. Further, those with bureau rating description indicating high-risk profile of customer showcase more defaulters while those indicating low risk profile showcase fewer defaulters.

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

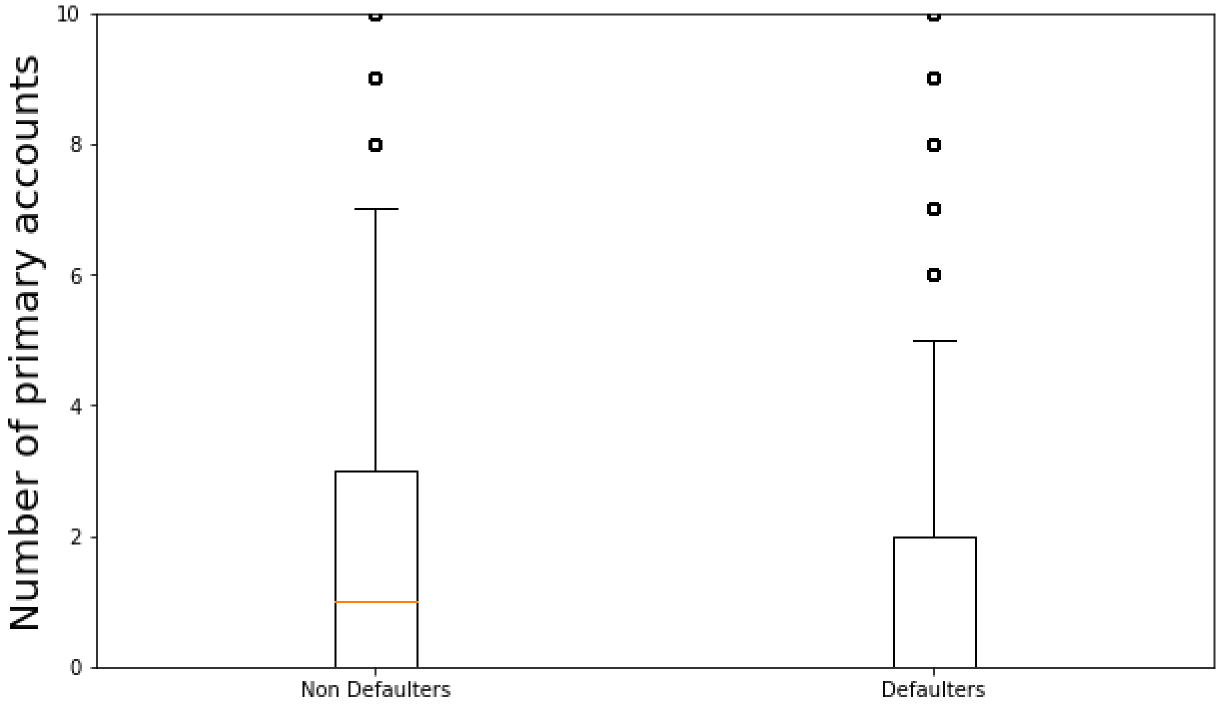
In [130…

```
fig37,ax37=plt.subplots(1,1,figsize=(10,6));
ax37.boxplot([loan_df.PRI_NO_OF_ACCTS[loan_df.loan_default==0],loan_df.PRI_NO_OF_ACCTS[loan_df.loan_default==1]]);
ax37.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax37.set_ylabel('Number of primary accounts',size=20);
```



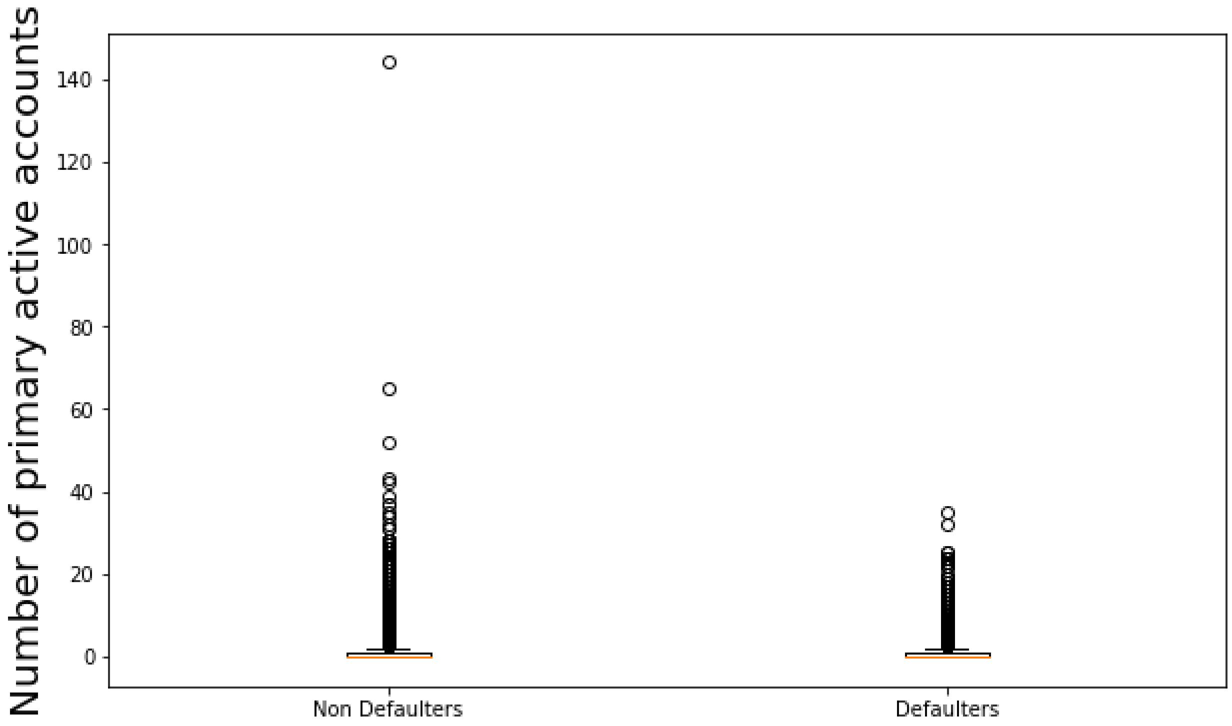
In [131…

```
fig38,ax38=plt.subplots(1,1,figsize=(10,6));
ax38.boxplot([loan_df.PRI_NO_OF_ACCTS[loan_df.loan_default==0],loan_df.PRI_NO_OF_ACCTS[loan_df.loan_default==1]]);
ax38.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax38.set_ylabel('Number of primary accounts',size=20);
ax38.set_ylim([0,10]);
```



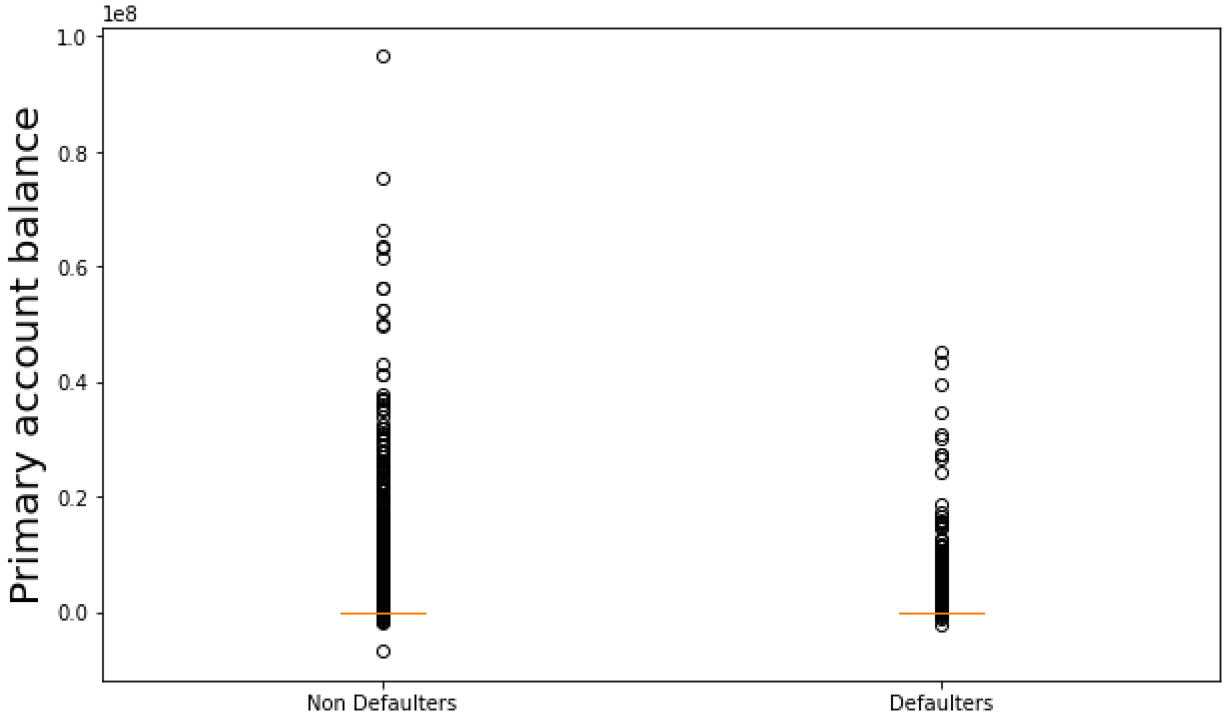
In [132…

```
fig39,ax39=plt.subplots(1,1,figsize=(10,6));
ax39.boxplot([loan_df.PRI_ACTIVE_ACCTS[loan_df.loan_default==0],loan_df.PRI_ACTIVE_ACCTS[loan_df.loan_default==1]]);
ax39.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax39.set_ylabel('Number of primary active accounts',size=20);
```



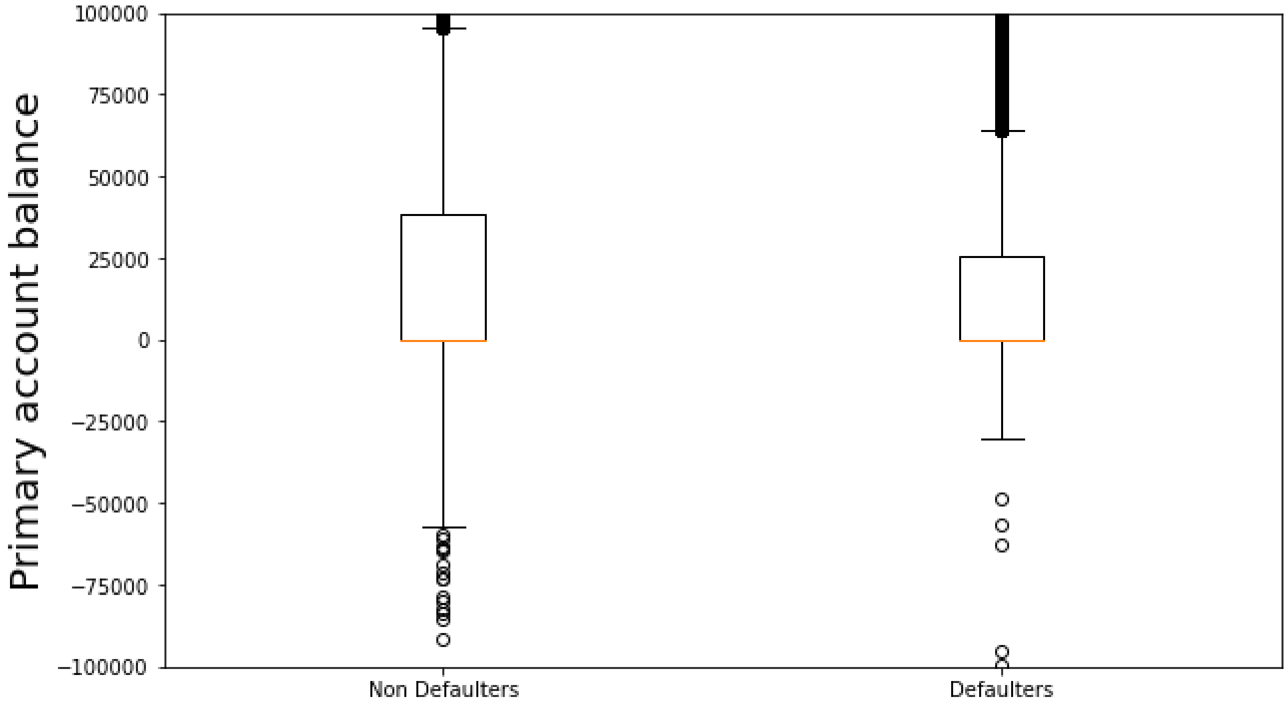
In [133…

```
fig40,ax40=plt.subplots(1,1,figsize=(10,6));
ax40.boxplot([loan_df.PRI_CURRENT_BALANCE[loan_df.loan_default==0],loan_df.PRI_CURRENT_BALANCE[loan_df.loan_default==1]]);
ax40.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax40.set_ylabel('Primary account balance',size=20);
```



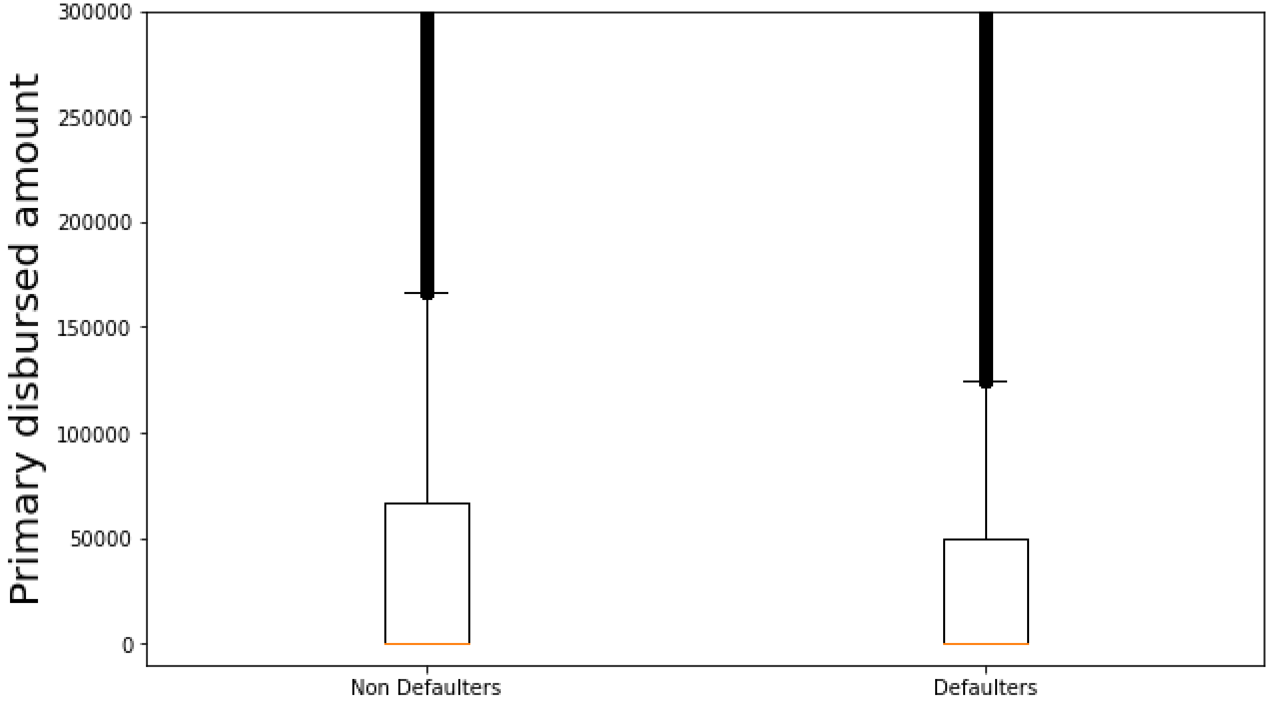
In [134...

```
fig41,ax41=plt.subplots(1,1,figsize=(10,6));
ax41.boxplot([loan_df.PRI_CURRENT_BALANCE[loan_df.loan_default==0],loan_df.PRI_CURRENT_BALANCE[loan_df.loan_default==1]]);
ax41.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax41.set_ylabel('Primary account balance',size=20);
ax41.set_ylim([-100000,100000]);
```



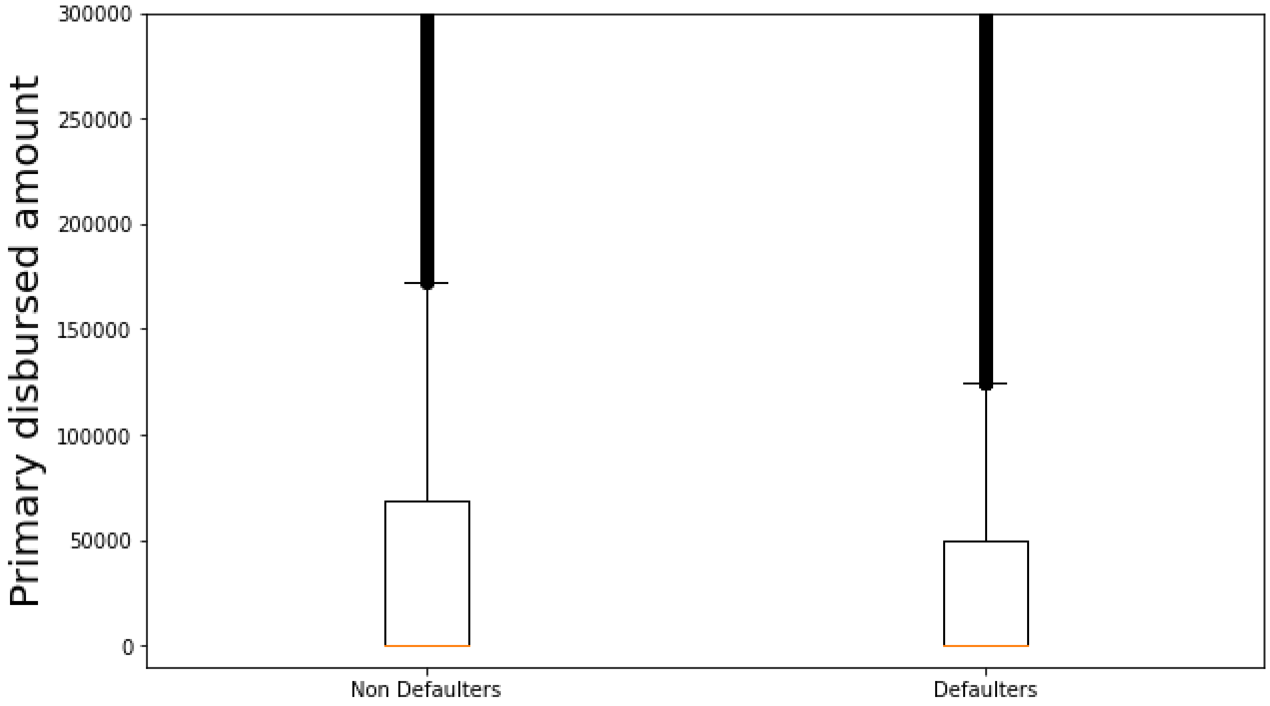
In [135…

```
fig42,ax42=plt.subplots(1,1,figsize=(10,6));
ax42.boxplot([loan_df.PRI_DISBURSED_AMOUNT[loan_df.loan_default==0],loan_df.PRI_DISBURSED_AMOUNT[loan_df.loan_default==1]]);
ax42.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax42.set_ylabel('Primary disbursed amount',size=20);
ax42.set_ylim([-10000,300000]);
```



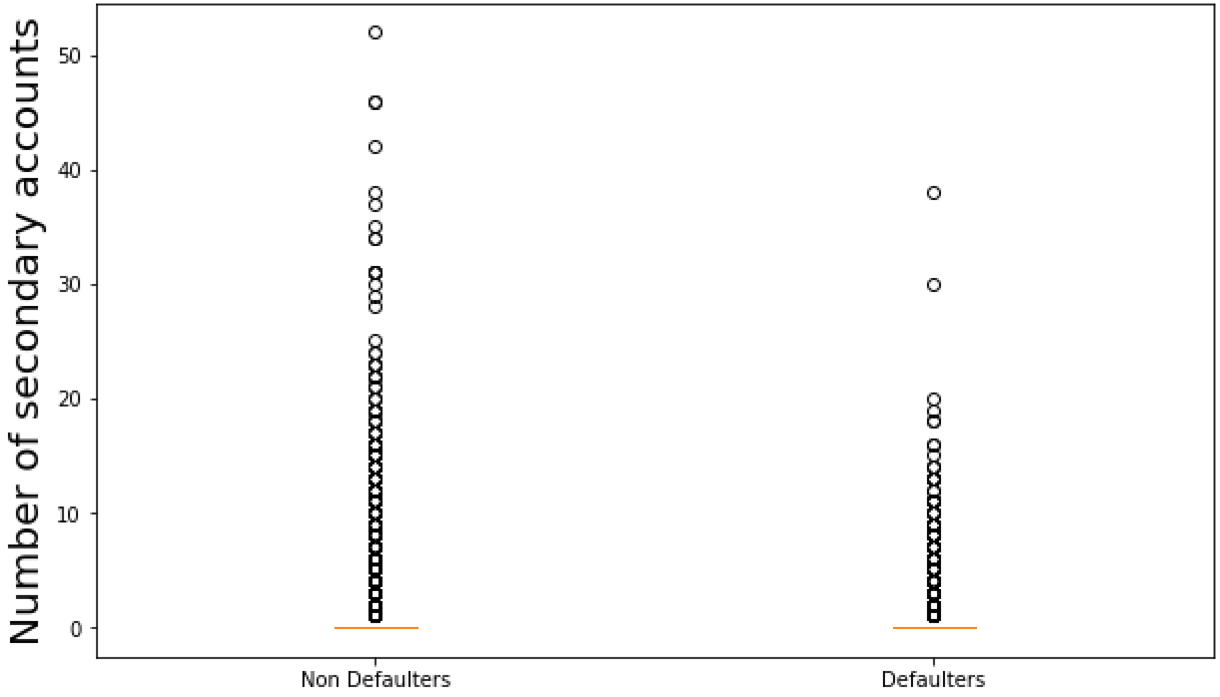
In [136…

```
fig43,ax43=plt.subplots(1,1,figsize=(10,6));
ax43.boxplot([loan_df.PRI_SANCTIONED_AMOUNT[loan_df.loan_default==0],loan_df.PRI_SANCTIONED_AMOUNT[loan_df.loan_default==1]]);
ax43.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax43.set_ylabel('Primary disbursed amount',size=20);
ax43.set_ylim([-10000,300000]);
```



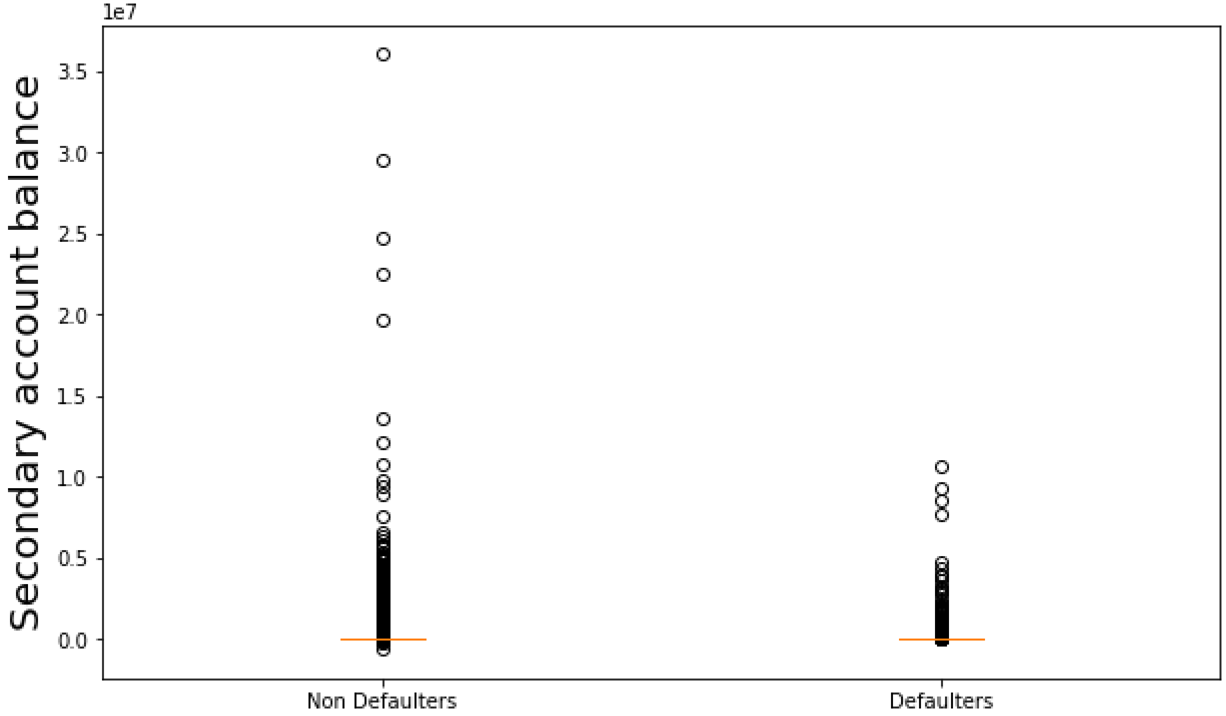
In [137…

```
fig44,ax44=plt.subplots(1,1,figsize=(10,6));
ax44.boxplot([loan_df.SEC_NO_OF_ACCTS[loan_df.loan_default==0],loan_df.SEC_NO_OF_ACCTS[loan_df.loan_default==1]]);
ax44.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax44.set_ylabel('Number of secondary accounts',size=20);
```



In [138…

```
fig45,ax45=plt.subplots(1,1,figsize=(10,6));
ax45.boxplot([loan_df.SEC_CURRENT_BALANCE[loan_df.loan_default==0],loan_df.SEC_CURRENT_BALANCE[loan_df.loan_default==1]]);
ax45.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax45.set_ylabel('Secondary account balance',size=20);
```



In [ ]:

In [ ]:

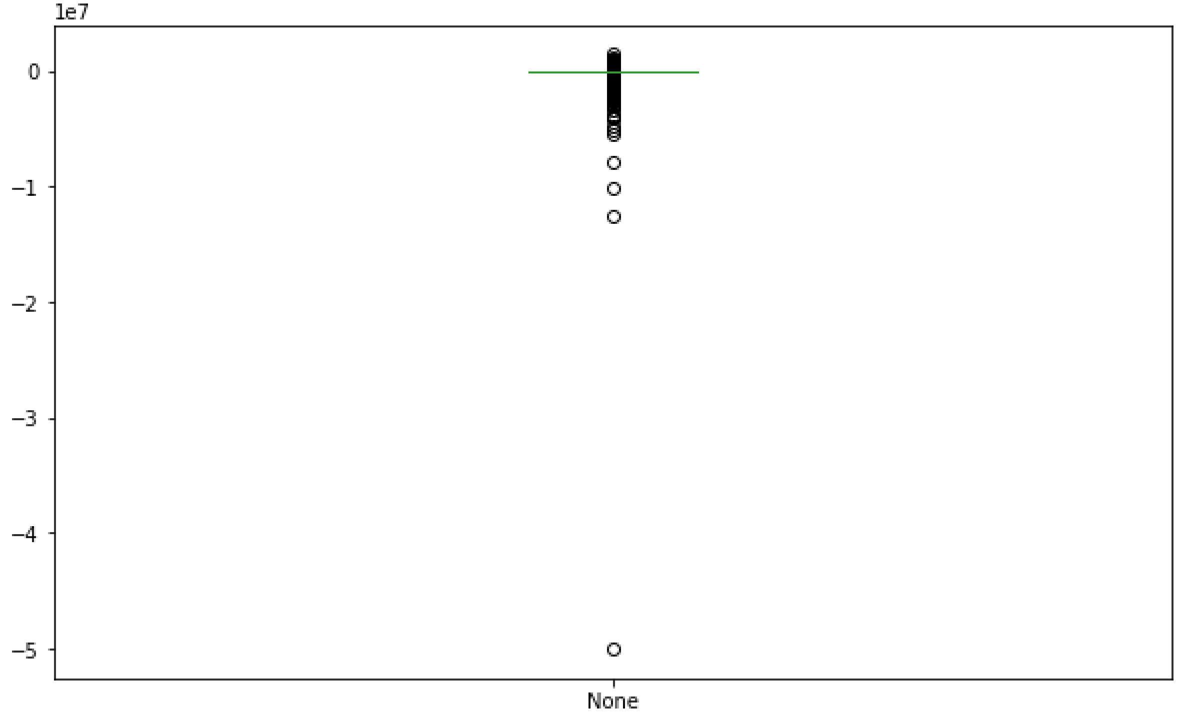
If we ignore few extreme outliers, non-defaulters tend to have more number of primary accounts with higher balance and are sanctioned and disbursed with greater amount compared to defaulters.

Most customers (both defaulter and non-defaulter) do not have any secondary account. In case of few outliers, non-defaulters tend to have more number of secondary accounts with greater account balance

3. Is there a difference between the sanctioned and disbursed amount of primary and secondary loans? Study the difference by providing appropriate statistics and graphs.

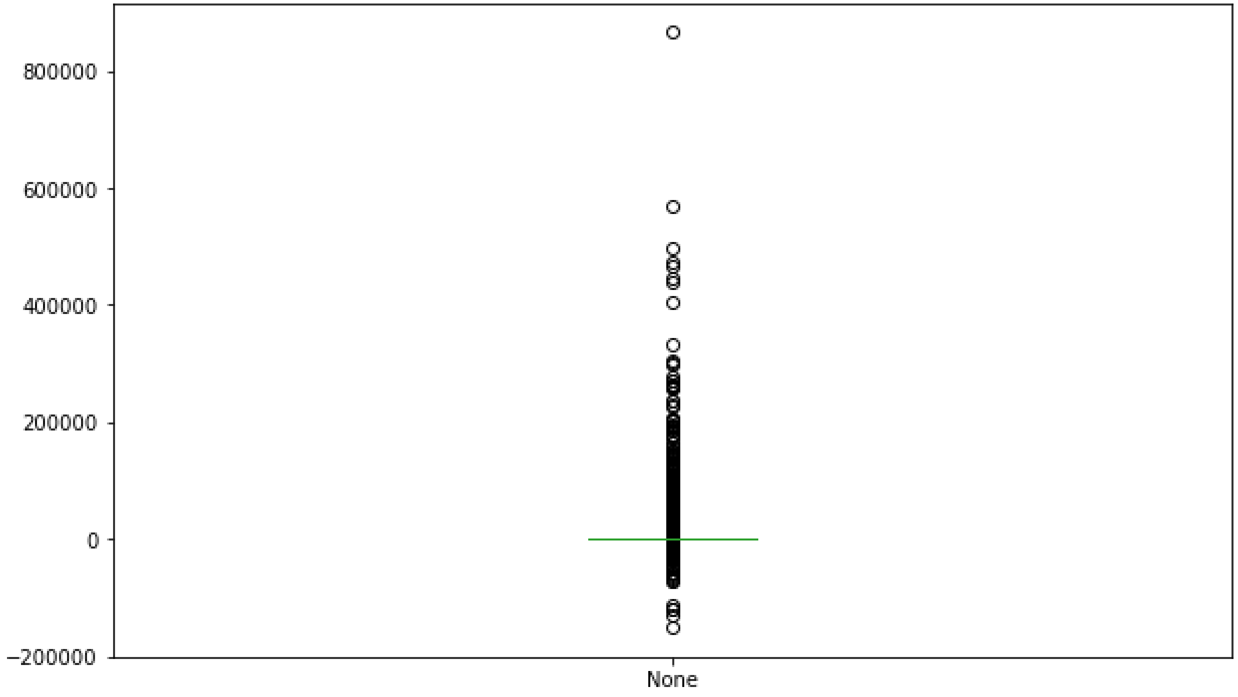
In [139…

```
fig46,ax46=plt.subplots(1,1,figsize=(10,6));
(loan_df.PRI_SANCTIONED_AMOUNT-loan_df.PRI_DISBURSED_AMOUNT).plot.box(ax=ax46);
```



In [140…

```
fig47,ax47=plt.subplots(1,1,figsize=(10,6));
(loan_df.SEC_SANCTIONED_AMOUNT-loan_df.SEC_DISBURSED_AMOUNT).plot.box(ax=ax47);
```

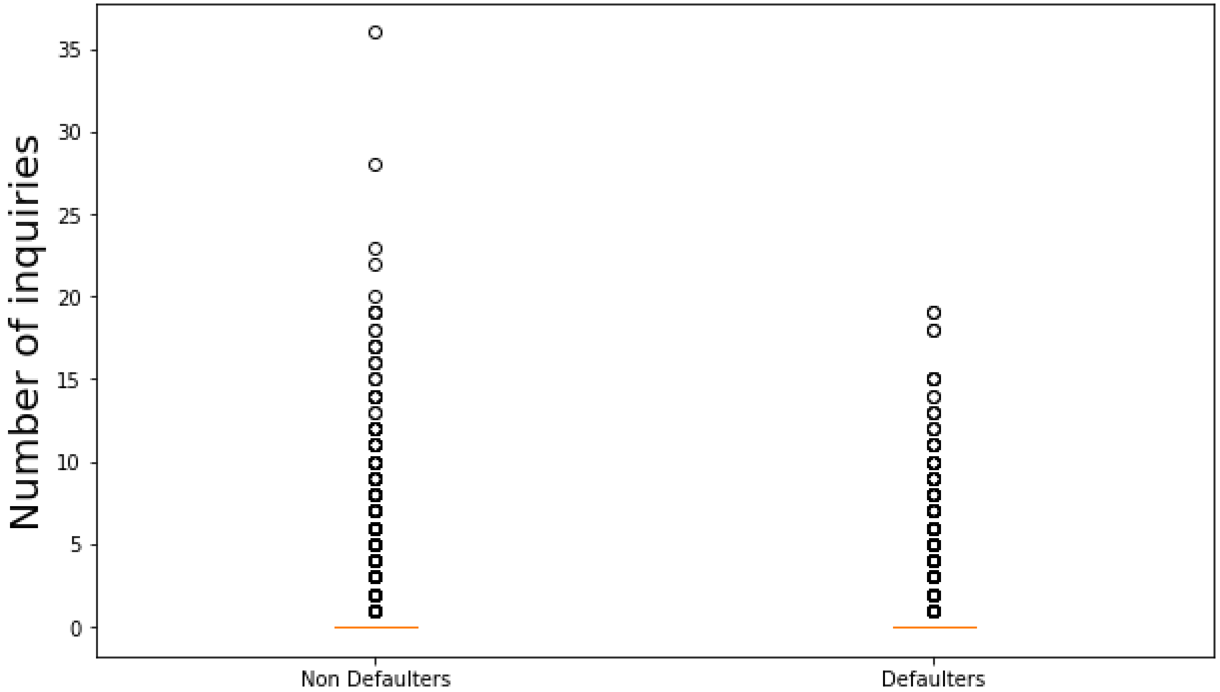


Though there is no difference between sanctioned and disbused amount for primary and secondary accounts of most customers, significant difference is observed for few customers.

4. Do customer who make higher number of enquiries end up being higher risk candidates?

In [141…

```
fig48,ax48=plt.subplots(1,1,figsize=(10,6));
ax48.boxplot([loan_df.NO_OF_INQUIRIES[loan_df.loan_default==0],loan_df.NO_OF_INQUIRIES[loan_df.loan_default==1]]);
ax48.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax48.set_ylabel('Number of inquiries',size=20);
```



In [142…

```
max(loan_df.NO_OF_INQUIRIES[loan_df.loan_default==0])
```

Out[142… 36

In [143…

```
max(loan_df.NO_OF_INQUIRIES[loan_df.loan_default==1])
```

Out[143… 19

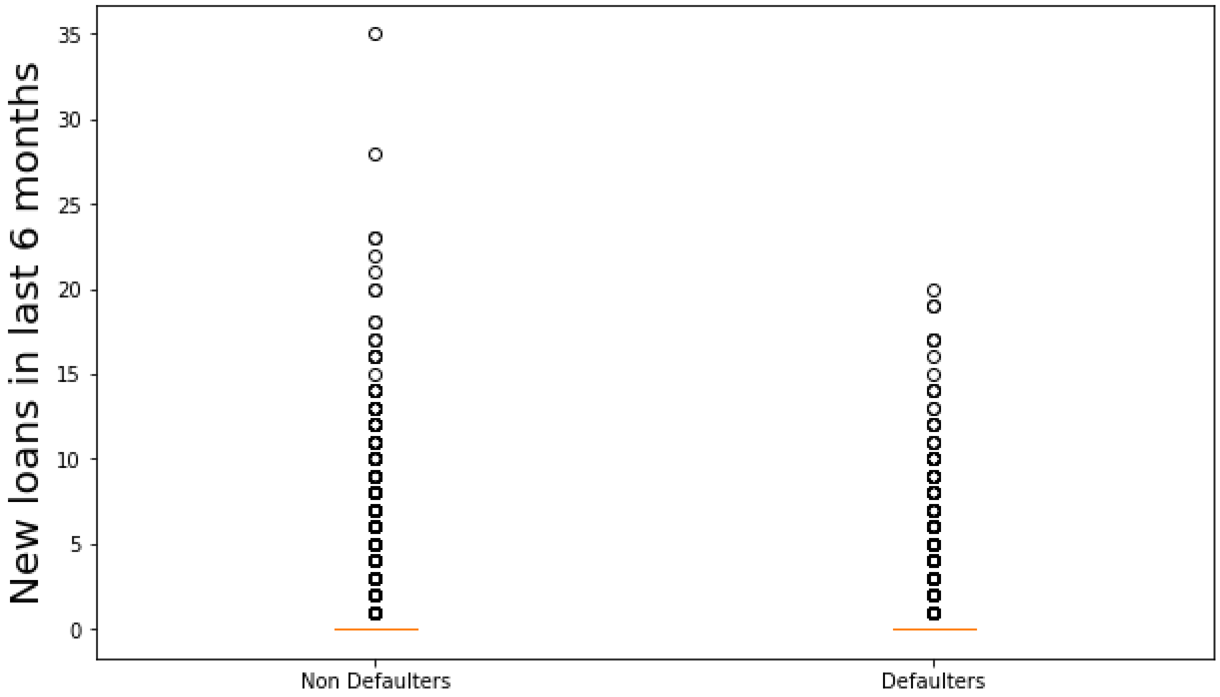
Most customers do not make any inquiry. Among outliers, few defaulters and non-defaulter make inquires. However, only few non-defaulters in the data analyzed make more than 19 inquires.

5. Is credit history, that is new loans in last six months, loans defaulted in last six months, time since first loan, etc., a significant factor in estimating probability of loan defaulters?

In [144…

```
fig49,ax49=plt.subplots(1,1,figsize=(10,6));
ax49.boxplot([loan_df.NEW_ACCTS_IN_LAST_SIX_MONTHS[loan_df.loan_default==0],loan_df.NEW_ACCTS_IN_LAST_SIX_MONTHS[loan_df.loan_defa
```

```
ax49.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax49.set_ylabel('New loans in last 6 months',size=20);
```



In [145…

```
new_loan_non_def=loan_df.NEW_ACCTS_IN_LAST_SIX_MONTHS[loan_df.loan_default==0].value_counts();
```

In [146…

```
new_loan_def=loan_df.NEW_ACCTS_IN_LAST_SIX_MONTHS[loan_df.loan_default==1].value_counts();
```

In [147…

```
new_loan_dist=pd.merge(new_loan_non_def,new_loan_def ,how='outer',left_index=True,right_index=True);
```

In [148…

```
new_loan_dist.fillna(0,inplace=True);
new_loan_dist.rename(columns={'NEW_ACCTS_IN_LAST_SIX_MONTHS_x':'num_non_defaulters','NEW_ACCTS_IN_LAST_SIX_MONTHS_y':'num_defaulte
new_loan_dist.sort_index(inplace=True)
```

In [149…

```
new_loan_dist['defaulter_ratio']=new_loan_dist.num_defaulters/(new_loan_dist.num_non_defaulters+new_loan_dist.num_defaulters);
```

```
new_loan_dist.sort_values(by=['defaulter_ratio'],ascending=False,inplace=True)
```

In [150…

```
new_loan_dist
```

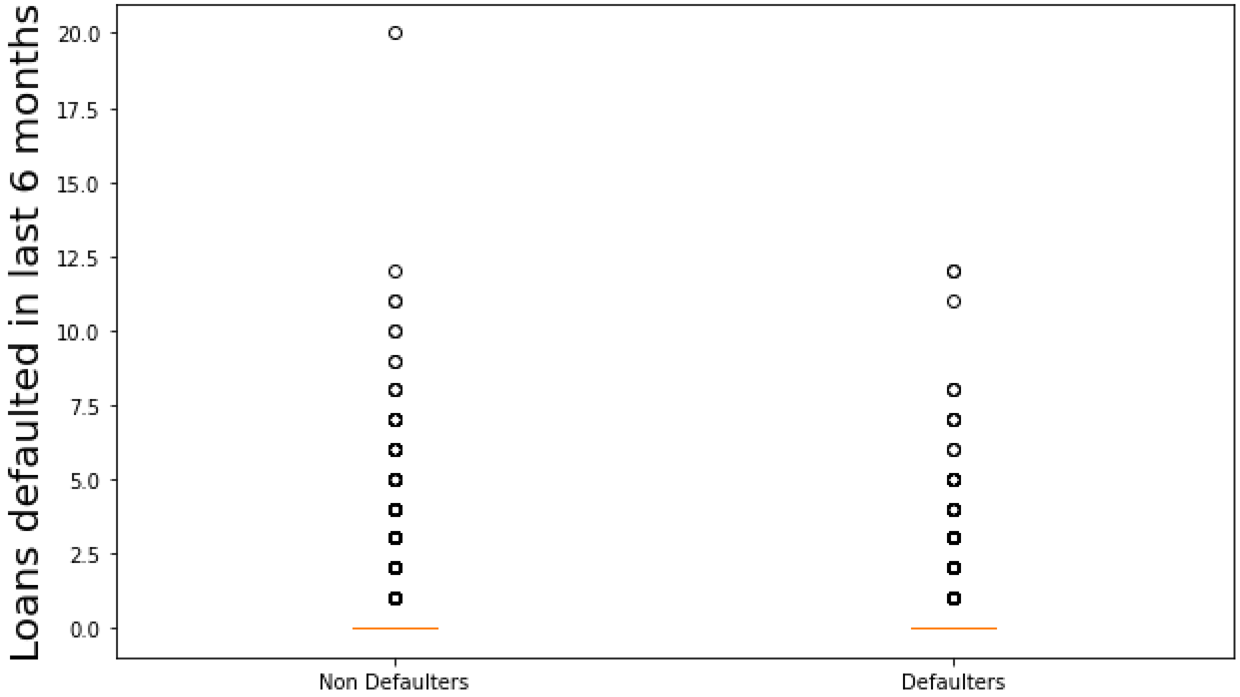
Out[150…

	num_non_defaulters	num_defaulters	defaulter_ratio
19	0.0	2.0	1.000000
17	3.0	3.0	0.500000
15	1.0	1.0	0.500000
14	7.0	4.0	0.363636
20	2.0	1.0	0.333333
12	15.0	5.0	0.250000
0	140812.0	40682.0	0.224151
10	43.0	12.0	0.218182
1	25735.0	6364.0	0.198262
2	8931.0	2084.0	0.189197
4	1609.0	348.0	0.177823
3	3690.0	768.0	0.172275
6	398.0	82.0	0.170833
5	800.0	164.0	0.170124
16	5.0	1.0	0.166667
8	123.0	24.0	0.163265
11	26.0	5.0	0.161290
7	255.0	47.0	0.155629
9	67.0	12.0	0.151899
13	13.0	2.0	0.133333
18	2.0	0.0	0.000000

	num_non_defaulters	num_defaulters	defaulter_ratio
21	1.0	0.0	0.000000
22	1.0	0.0	0.000000
23	2.0	0.0	0.000000
28	1.0	0.0	0.000000
35	1.0	0.0	0.000000

In [151…

```
fig50,ax50=plt.subplots(1,1,figsize=(10,6));
ax50.boxplot([loan_df.DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS[loan_df.loan_default==0],loan_df.DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS[loa
ax50.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax50.set_ylabel('Loans defaulted in last 6 months',size=20);
```



In [152…

```
delinq_non_def=loan_df.DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS[loan_df.loan_default==0].value_counts();
delinq_def=loan_df.DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS[loan_df.loan_default==1].value_counts();
```

```
deliq_dist=pd.merge(delinq_non_def,delinq_def ,how='outer',left_index=True,right_index=True);
```

In [153…

```
delinq_dist.fillna(0,inplace=True);
delinq_dist.rename(columns={'DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS_x':'num_non_defaulters','DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS_y':'
delinq_dist.sort_index(inplace=True)
```

In [154…

```
delinq_dist['defaulter_ratio']=delinq_dist.num_defaulters/(delinq_dist.num_non_defaulters+delinq_dist.num_defaulters);
delinq_dist.sort_values(by=['defaulter_ratio'],ascending=False,inplace=True)
```

In [155…

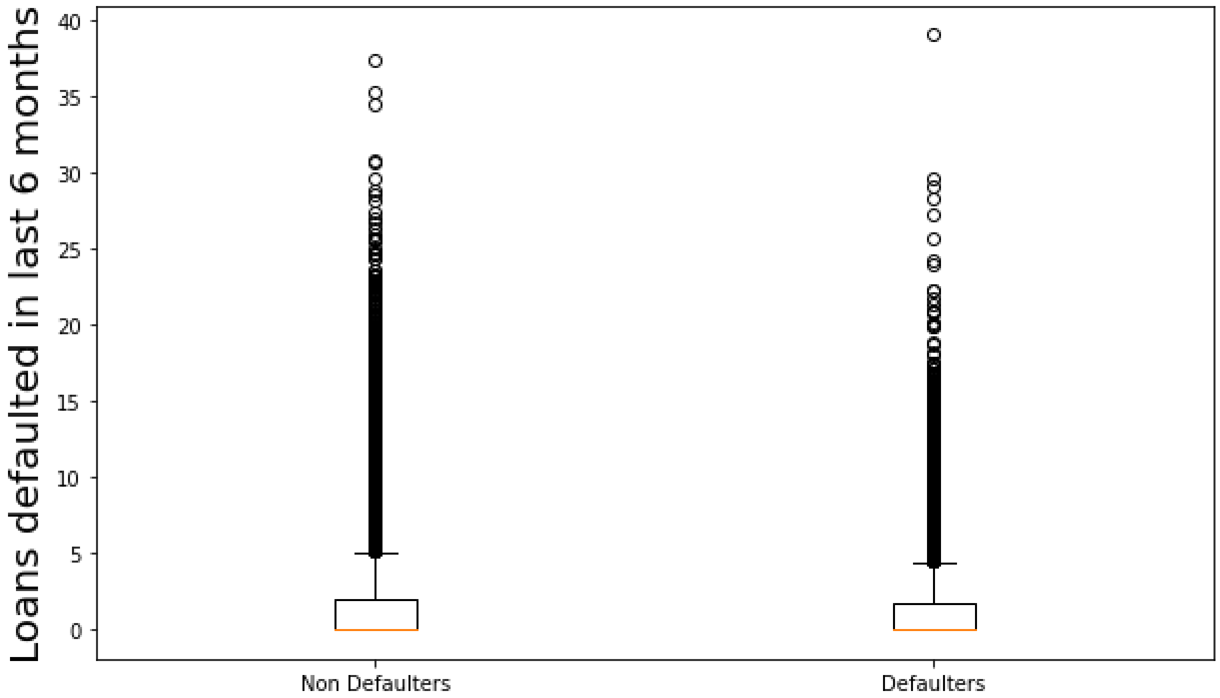
```
delinq_dist
```

Out[155…

	num_non_defaulters	num_defaulters	defaulter_ratio
12	1	2.0	0.666667
8	4	3.0	0.428571
7	8	5.0	0.384615
11	2	1.0	0.333333
4	96	42.0	0.304348
3	385	152.0	0.283054
2	1784	686.0	0.277733
5	42	16.0	0.275862
1	10922	4019.0	0.268991
0	169277	45682.0	0.212515
6	17	3.0	0.150000
9	2	0.0	0.000000
10	2	0.0	0.000000
20	1	0.0	0.000000

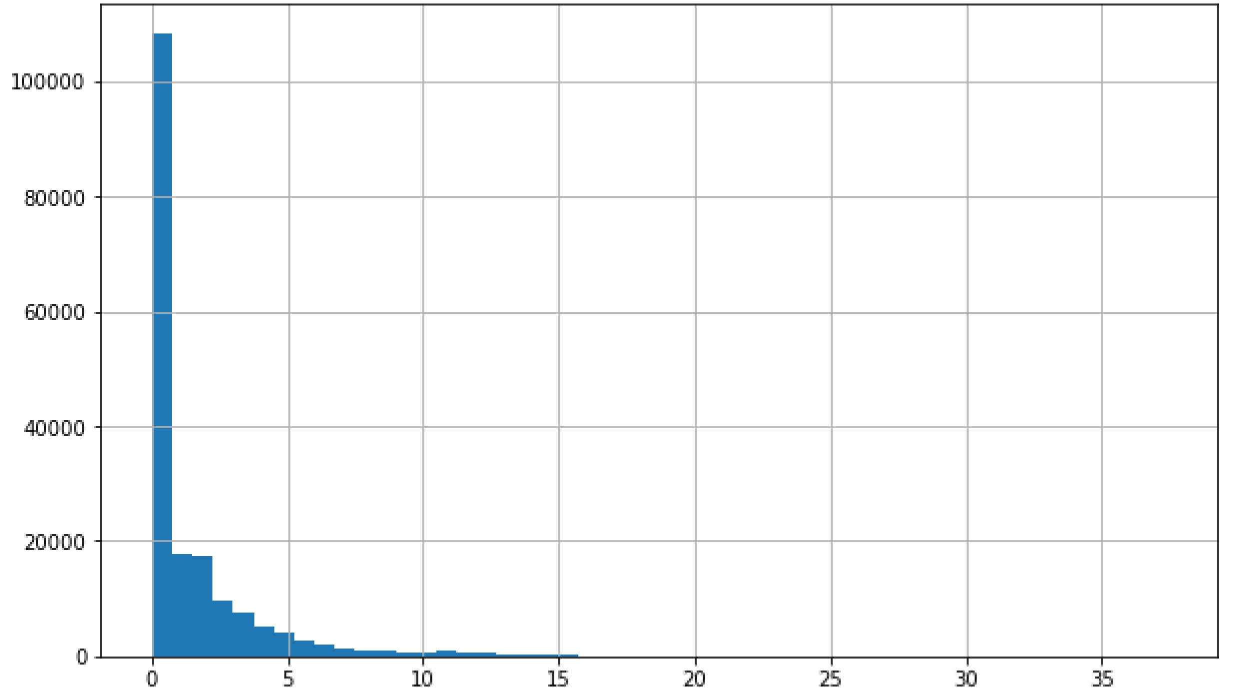
In [156…

```
fig51,ax51=plt.subplots(1,1,figsize=(10,6));
ax51.boxplot([loan_df.CREDIT_HISTORY_LENGTH[loan_df.loan_default==0],loan_df.CREDIT_HISTORY_LENGTH[loan_df.loan_default==1]]);
ax51.set_xticklabels(['Non Defaulters','Defaulters'],size=10);
ax51.set_ylabel('Loans defaulted in last 6 months',size=20);
```



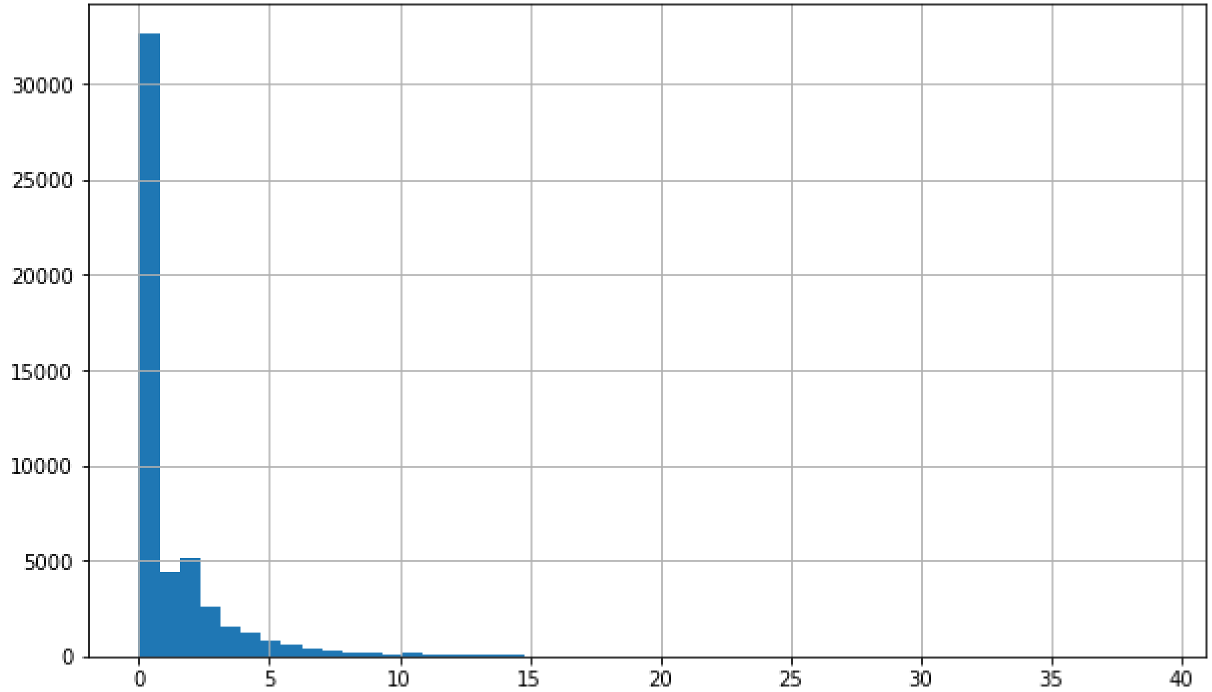
In [157…

```
fig52,ax52=plt.subplots(1,1,figsize=(10,6));
loan_df.CREDIT_HISTORY_LENGTH[loan_df.loan_default==0].hist(ax=ax52,bins=50);
```



In [158…

```
fig53,ax53=plt.subplots(1,1,figsize=(10,6));
loan_df.CREDIT_HISTORY_LENGTH[loan_df.loan_default==1].hist(ax=ax53,bins=50);
```



In [ ]:

In [159…

```
spearmanr(loan_df.NEW_ACCTS_IN_LAST_SIX_MONTHS,loan_df.loan_default)
```

Out[159…

```
SpearmanrResult(correlation=-0.033166575969647984, pvalue=9.393850315048903e-58)
```

In [160…

```
spearmanr(loan_df.DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS,loan_df.loan_default)
```

Out[160…

```
SpearmanrResult(correlation=0.03805781161461587, pvalue=1.7920328815785558e-75)
```

In [161…

```
spearmanr(loan_df.CREDIT_HISTORY_LENGTH,loan_df.loan_default)
```

Out[161…

```
SpearmanrResult(correlation=-0.04042643503420887, pvalue=6.323461808906134e-85)
```

There is not significant correlation of new loans in last six month, loans defaulted in last 6 months and time since first

loan and propability of defaulting`

However, this negligible correlation is observed because most customers have zero or very few new loan in 6 months and have zero loan defaults in 6 months.

If we only observe outliers separately, it is noticed that many customers who have new loans in last six months are non-defaulters. Similarly, many defaulters have loans defaulted in last 6 months.

Observations based on Week 2 Questions 1-5

1. Boxplots show that non-defaulter have higher average bureau score compared to defaulters.
2. Further, those with bureau rating description indicating high-risk profile of customer showcase more defaulters while those indicating low risk profile showcase fewer defaulters.
3. If we ignore few extreme outliers, non-defaulters tend to have more number of primary accounts with higher balance and are sanctioned and disbursed with greater amount compared to defaulters.
4. Most customers (both defaulter and non-defaulter) do not have any secondary account. In case of few outliers, non-defaulters tend to have more number of secondary accounts with greater account balance
5. Though there is no difference between sanctioned and disbused amount for primary and secondary accounts of most customers, significant difference is observed for few customers.
6. Most customers do not make any inquiry. Among outliers, few defaulters and non-defaulter make inquires. However, only few non-defaulters in the data analyzed make more than 19 inquires. None of the defaulters makes more than 19 inquiries.
7. There is not significant correlation of new loans in last six month, loans defaulted in last 6 months and time since first loan and propability of defaulting`
8. However, this negligible correlation is observed because most customers have zero or very few new loan in 6 months and have zero loan defaults in 6 months.
9. If we only observe outliers separately, it is noticed that many customers who have new loans in last six months are non-defaulters. Similarly, many defaulters have loans defaulted in last 6 months.

In [ ]:

```

```

6. Perform logistic regression modeling, predict the outcome for the test data, and validate the results using the confusion matrix.

In [162…

```
loan_df
```

Out[162…

	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	Salaried	...
1	417566	53278	61360	89.63	67	22807	45	1497	1985-08-24	Self employed	...
2	539055	52378	60300	88.39	67	22807	45	1495	1977-12-09	Self employed	...
3	529269	46349	61500	76.42	67	22807	45	1502	1988-06-01	Salaried	...
4	563215	43594	78256	57.50	67	22744	86	1499	1994-07-14	Self employed	...
...	...	...	...	...	...	...	...	...	...	...	...
233149	561031	57759	76350	77.28	5	22289	51	3326	1981-11-10	Self employed	...
233150	649600	55009	71200	78.72	138	17408	51	3385	1992-10-15	Self employed	...
233151	603445	58513	68000	88.24	135	23313	45	1797	1981-12-19	Self employed	...
233152	442948	22824	40458	61.79	160	16212	48	96	1989-07-31	Self employed	...
233153	545300	35299	72698	52.27	3	14573	45	17	1968-08-01	Self employed	...

233154 rows × 42 columns

In [163…

```
loan_df.columns
```

Out[163…

```
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', 'age'],
      dtype='object')
```

In [164…

```
np.shape(loan_df)
```

Out[164…

```
(233154, 42)
```

In [165…

```
# Drop
# UniqueID
# Date_of_Birth
# DisbursalDate
# MobileNo_Avl_Flag
# PERFORM_CNS_SCORE
```

In [277…

```
loan_df_2=loan_df.drop(columns=['UniqueID','Date_of_Birth','DisbursalDate','MobileNo_Avl_Flag','PERFORM_CNS_SCORE','Current_pincod
```

In [278…

```
np.shape(loan_df_2)
```

Out[278…

```
(233154, 35)
```

In [279…

```
# get_dummies
# branch_id
# manufacturer_id
# Employment_Type
# State_ID
# PERFORM_CNS_SCORE_DESCRIPTION
```

In [280…

```
loan_df_2_dummies=pd.get_dummies(loan_df_2[['branch_id','manufacturer_id','Employment_Type','State_ID','PERFORM_CNS_SCORE_DESCRIPT
```

In [281…

```
np.shape(loan_df_2_dummies)
```

Out[281…

```
(233154, 132)
```

In [282…

```
loan_df_2=pd.merge(loan_df_2,loan_df_2_dummies ,how='left',left_index=True,right_index=True);
loan_df_2=loan_df_2.drop(columns=['branch_id','supplier_id','manufacturer_id','Employment_Type','State_ID','PERFORM_CNS_SCORE_DESC
```

In [283…

```
np.shape(loan_df_2)
```

Out[283…

```
(233154, 161)
```

In [284…

```
spearmanr(loan_df.disbursed_amount,loan_df.asset_cost)
```

Out[284…

```
SpearmanrResult(correlation=0.670569337078484, pvalue=0.0)
```

In [285…

```
spearmanr(loan_df.disbursed_amount,loan_df.ltv)
```

Out[285…

```
SpearmanrResult(correlation=0.4222823320162862, pvalue=0.0)
```

Since disbursed_amount, asset_cost and ltv are strongly correlated, we drop the asset_cost and ltv column

In [287…

```
loan_df_2=loan_df_2.drop(columns=['asset_cost','ltv'])
```

In [288…

```
loan_df_train,loan_df_test=train_test_split(loan_df_2,test_size=0.15,random_state=0);
```

In [289…

```
np.shape(loan_df_train)
```

Out[289…

```
(198180, 159)
```

In [290…

```
np.shape(loan_df_test)
```

Out[290…

```
(34974, 159)
```

In [291…

```
X_train=loan_df_train.drop(columns=['loan_default']);
y_train=loan_df_train.loan_default;
```

```
X_test=loan_df_test.drop(columns=['loan_default']);
y_test=loan_df_test.loan_default;
```

In [292…

```
print(np.shape(X_train))
print(np.shape(y_train))
print(np.shape(X_test))
print(np.shape(y_test))
```

```
(198180, 158)
(198180,)
(34974, 158)
(34974,)
```

In [296…

```
transformer = Normalizer().fit(X_train[['disbursed_amount','PRI_NO_OF_ACCTS','PRI_ACTIVE_ACCTS','PRI_OVERDUE_ACCTS', 'PRI_CURRENT_
```

In [297…

```
X_train[['disbursed_amount','PRI_NO_OF_ACCTS','PRI_ACTIVE_ACCTS','PRI_OVERDUE_ACCTS', 'PRI_CURRENT_BALANCE', 'PRI_SANCTIONED_AMOUN
```

In [298…

```
X_test[['disbursed_amount','PRI_NO_OF_ACCTS','PRI_ACTIVE_ACCTS','PRI_OVERDUE_ACCTS', 'PRI_CURRENT_BALANCE', 'PRI_SANCTIONED_AMOUNT
```

In [469…

```
lr_model=LogisticRegression(C=0.1,penalty='elasticnet',solver='saga',l1_ratio=0.75);
```

In [470…

```
lr_model.fit(X_train,y_train)
```

Out[470…

```
▾ LogisticRegression
LogisticRegression(C=0.1, l1_ratio=0.75, penalty='elasticnet', solver='saga')
```

In [471…

```
lr_model.score(X_train,y_train)
```

Out[471…

```
0.7827934201231204
```

In [472…

```
lr_model.score(X_test,y_test)
```

Out[472…
```
0.7829816435066049
```

In [473…
```
lr_model.intercept_
```

Out[473…
```
array([-2.93510838])
```

In [474…
```
lr_model.coef_
```

Out[474…
```
array([[ 1.44296613, -0.08066312, -0.09834279,  0.06541072, -0.12572076,
        -0.28048665,  0.        ,  0.        ,  0.        ,  0.76135884,
        -0.14380981,  1.18183296,  0.        ,  0.        ,  0.        ,
         0.        ,  0.75009289,  0.48603835,  0.26027867,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.64091899, -0.21558731,  0.        , -0.17003994,
        -0.1810993 ,  0.35274772,  0.27520353,  0.        ,  0.23823379,
         0.0914424 , -0.01469813,  0.        ,  0.        , -0.12353892,
        -0.09011626, -0.0227036 ,  0.13784633,  0.        ,  0.41143013,
         0.18992182, -0.12541186, -0.13916181,  0.33541139,  0.        ,
         0.        ,  0.41301182, -0.10509036, -0.22507407,  0.07506498,
        -0.32292859,  0.03910454, -0.28207297, -0.13156282, -0.07650713,
         0.        ,  0.        ,  0.        ,  0.21102777,  0.        ,
         0.04297638,  0.16733534,  0.15558179,  0.30090079,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.29326128,
        -0.21020513, -0.13739119,  0.02857908,  0.        ,  0.16135986,
        -0.0341232 , -0.05690134,  0.07382186,  0.08287561,  0.        ,
         0.        ,  0.1290473 , -0.04297432, -0.06246636, -0.04166376,
        -0.03073767,  0.05302329,  0.31076437, -0.07989087, -0.04527969,
         0.        ,  0.27895563,  0.        , -0.0263836 ,  0.32025269,
         0.00269217, -0.06791212, -0.05508942,  0.        ,  0.17426626,
         0.26824068,  0.27067321, -0.10322147,  0.        ,  0.        ,
         0.        ,  0.03138814,  0.12979511,  0.        , -0.03689982,
        -0.01270142, -0.08482698,  0.16555827, -0.15352177,  0.        ,
         0.0914424 ,  0.30514061, -0.01700545,  0.1058064 , -0.08455253,
         0.05616811,  0.05073844, -0.05508942,  0.18992182,  0.        ,
         0.        ,  0.        , -0.45806159,  0.03602164, -0.05821605,
         0.        ,  0.        ,  0.19002581,  0.        , -0.26162752,
        -0.09342347, -0.140473  ,  0.03630321,  0.11443741,  0.1872816 ,
         0.41813375,  0.58803715,  0.55483414,  0.69796783,  0.60496215,
         0.74879739,  0.24178285,  0.        ,  0.01630528,  0.        ,
         0.023551  ,  0.22534659,  0.27167041]])
```

In [475…
```
lr_model.feature_names_in_
```

Out[475…
```
array(['disbursed_amount', 'Aadhar_flag', 'PAN_flag', 'VoterID_flag',
       'Driving_flag', 'Passport_flag', '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', 'age', 'branch_id_10',
       'branch_id_100', 'branch_id_101', 'branch_id_103', 'branch_id_104',
       'branch_id_105', 'branch_id_11', 'branch_id_111', 'branch_id_117',
       'branch_id_120', 'branch_id_121', 'branch_id_13', 'branch_id_130',
       'branch_id_135', 'branch_id_136', 'branch_id_138', 'branch_id_14',
       'branch_id_142', 'branch_id_146', 'branch_id_147', 'branch_id_15',
       'branch_id_152', 'branch_id_153', 'branch_id_158', 'branch_id_159',
       'branch_id_16', 'branch_id_160', 'branch_id_162', 'branch_id_165',
       'branch_id_17', 'branch_id_18', 'branch_id_19', 'branch_id_2',
       'branch_id_20', 'branch_id_202', 'branch_id_207', 'branch_id_217',
       'branch_id_248', 'branch_id_249', 'branch_id_250', 'branch_id_251',
       'branch_id_254', 'branch_id_255', 'branch_id_257', 'branch_id_258',
       'branch_id_259', 'branch_id_260', 'branch_id_261', 'branch_id_29',
       'branch_id_3', 'branch_id_34', 'branch_id_35', 'branch_id_36',
       'branch_id_42', 'branch_id_43', 'branch_id_48', 'branch_id_5',
       'branch_id_61', 'branch_id_62', 'branch_id_63', 'branch_id_64',
       'branch_id_65', 'branch_id_66', 'branch_id_67', 'branch_id_68',
       'branch_id_69', 'branch_id_7', 'branch_id_70', 'branch_id_72',
       'branch_id_73', 'branch_id_74', 'branch_id_76', 'branch_id_77',
       'branch_id_78', 'branch_id_79', 'branch_id_8', 'branch_id_82',
       'branch_id_84', 'branch_id_85', 'branch_id_9', 'branch_id_97',
       'manufacturer_id_145', 'manufacturer_id_152',
       'manufacturer_id_153', 'manufacturer_id_156', 'manufacturer_id_45',
       'manufacturer_id_48', 'manufacturer_id_49', 'manufacturer_id_51',
       'manufacturer_id_67', 'manufacturer_id_86',
       'Employment_Type_Self employed', 'State_ID_10', 'State_ID_11',
       'State_ID_12', 'State_ID_13', 'State_ID_14', 'State_ID_15',
       'State_ID_16', 'State_ID_17', 'State_ID_18', 'State_ID_19',
       'State_ID_2', 'State_ID_20', 'State_ID_21', 'State_ID_22',
       'State_ID_3', 'State_ID_4', 'State_ID_5', 'State_ID_6',
       'State_ID_7', 'State_ID_8', 'State_ID_9',
       'PERFORM_CNS_SCORE_DESCRIPTION_B-Very Low Risk',
```

```
       'PERFORM_CNS_SCORE_DESCRIPTION_C-Very Low Risk',
       'PERFORM_CNS_SCORE_DESCRIPTION_D-Very Low Risk',
       'PERFORM_CNS_SCORE_DESCRIPTION_E-Low Risk',
       'PERFORM_CNS_SCORE_DESCRIPTION_F-Low Risk',
       'PERFORM_CNS_SCORE_DESCRIPTION_G-Low Risk',
       'PERFORM_CNS_SCORE_DESCRIPTION_H-Medium Risk',
       'PERFORM_CNS_SCORE_DESCRIPTION_I-Medium Risk',
       'PERFORM_CNS_SCORE_DESCRIPTION_J-High Risk',
       'PERFORM_CNS_SCORE_DESCRIPTION_K-High Risk',
       'PERFORM_CNS_SCORE_DESCRIPTION_L-Very High Risk',
       'PERFORM_CNS_SCORE_DESCRIPTION_M-Very High Risk',
       'PERFORM_CNS_SCORE_DESCRIPTION_No Bureau History Available',
       'PERFORM_CNS_SCORE_DESCRIPTION_Not Scored: More than 50 active Accounts found',
       'PERFORM_CNS_SCORE_DESCRIPTION_Not Scored: No Activity seen on the customer (Inactive)',
       'PERFORM_CNS_SCORE_DESCRIPTION_Not Scored: No Updates available in last 36 months',
       'PERFORM_CNS_SCORE_DESCRIPTION_Not Scored: Not Enough Info available on the customer',
       'PERFORM_CNS_SCORE_DESCRIPTION_Not Scored: Only a Guarantor',
       'PERFORM_CNS_SCORE_DESCRIPTION_Not Scored: Sufficient History Not Available'],
      dtype=object)
```

In [476…

```
lr_model.score(X_train,y_train)
```

Out[476…

```
0.7827934201231204
```

In [477…

```
lr_model.score(X_test,y_test)
```

Out[477…

```
0.7829816435066049
```

In [479…

```
confusion_matrix(y_test,lr_model.predict(X_test))
```

Out[479…

```
array([[27371,     8],
       [ 7582,    13]], dtype=int64)
```

Observations based on Logistic Regression model. Week 2- Question 6

1. The logistic Regression model developed predicts the correct result 78.28 % times
2. On observing the coefficients of the Logistic regression model, it is found that whether the customer is defaulter majorly depends upon disbursed_amount, PRI_DISBURSED_AMOUNT, Bureau score description etc. among other features like branch id, state id, manufacturing id etc.
3. Customers with higher disbursed_amount tend to have higher chances of being a defaulter.

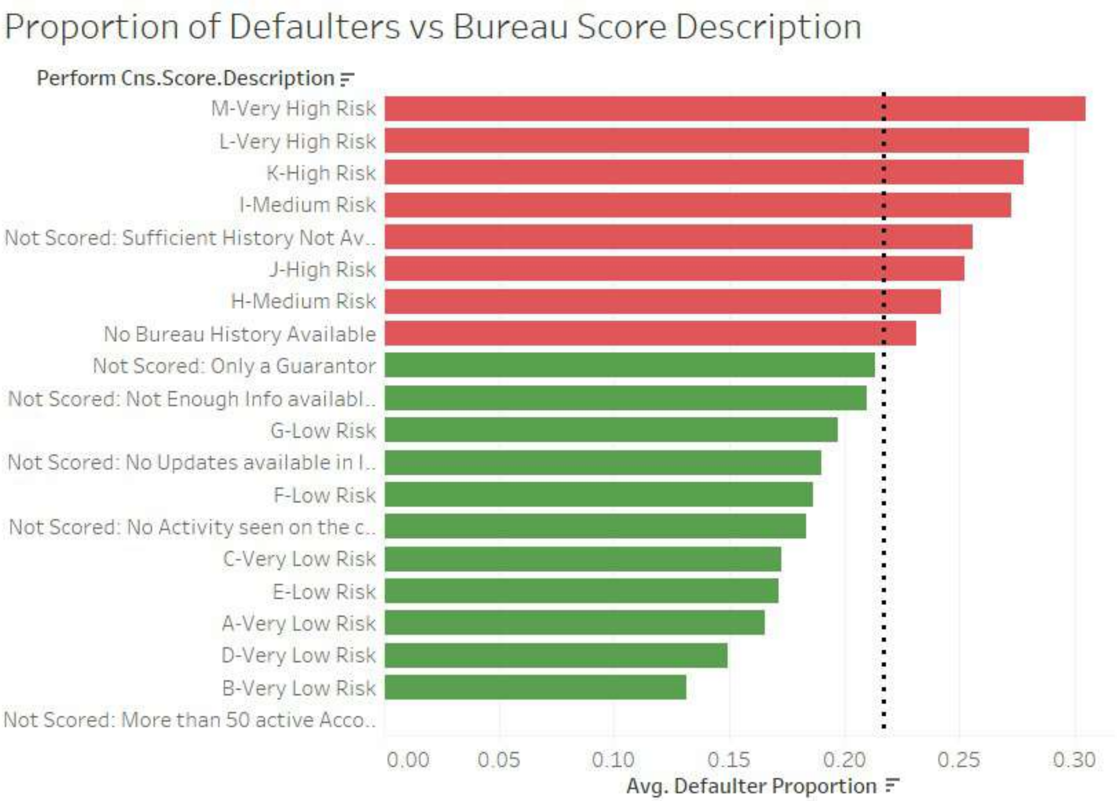
4. Customers with higher PRI_DISBURSED_AMOUNT tend to have higher chances of being a defaulter.
5. Customers with Very Low risk bureau Score have lower chances of being defaulters while customers Higher risk bureau score have greater chances of defaulting.
6. Based on the confusion matrix, it is observed that the model is able to identify non-defaulters more accurately but is not able to find sufficient number of defaulters. This implies that if the model predicts a defaulter, it can be suggested to not give loan to such candidate. However, if the model does not predict a defaulter, it is difficult to indicate whether the candidate will default or not.

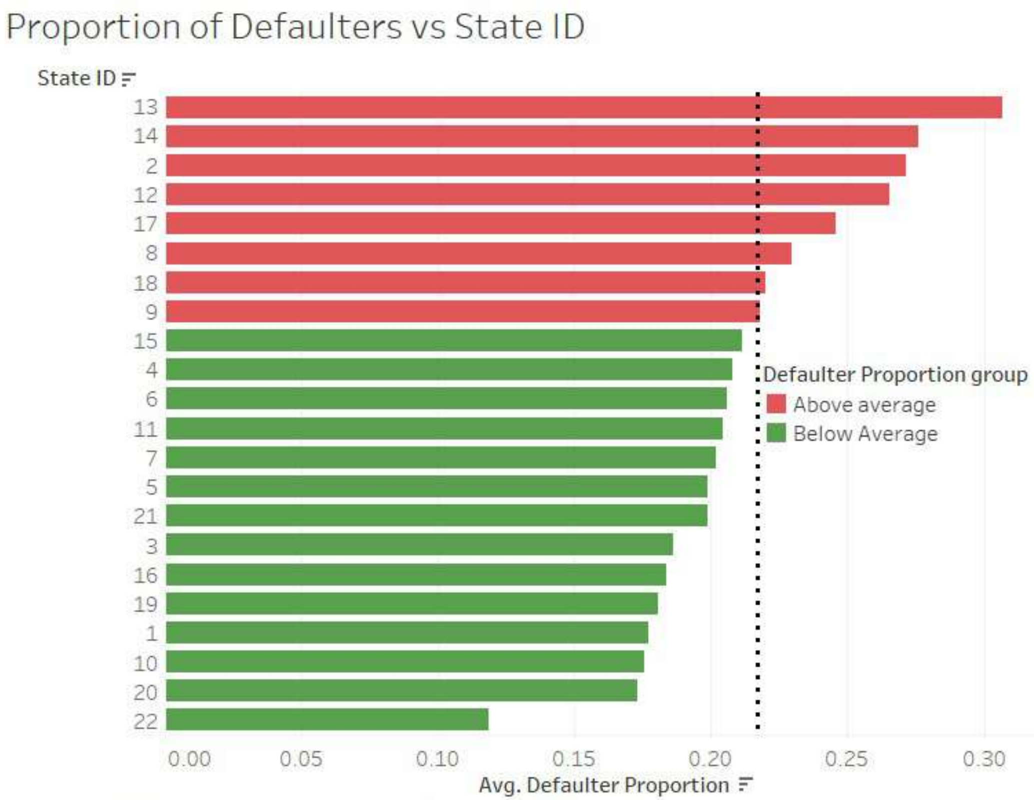
In [ ]:

In [ ]:

Bank Loan Defaulter Data Analysis

1/2





We can use Bureau Score and State information to estimate the possibility that a customer's loan will default.

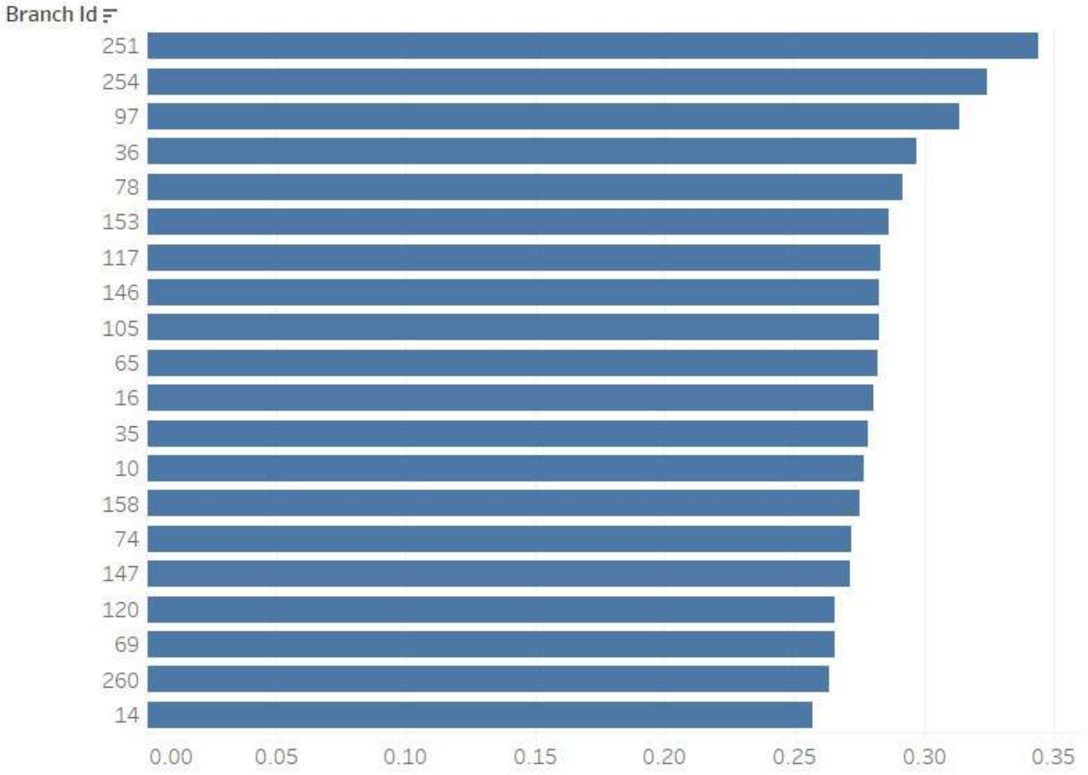
It is observed that more than average proportion of defaulters have higher risk Bureau Score Rating.

It is also observed that a very large proportion of defaulters are from States 13 and 14, while States 20 and 22 have very low proportion of defaulters.

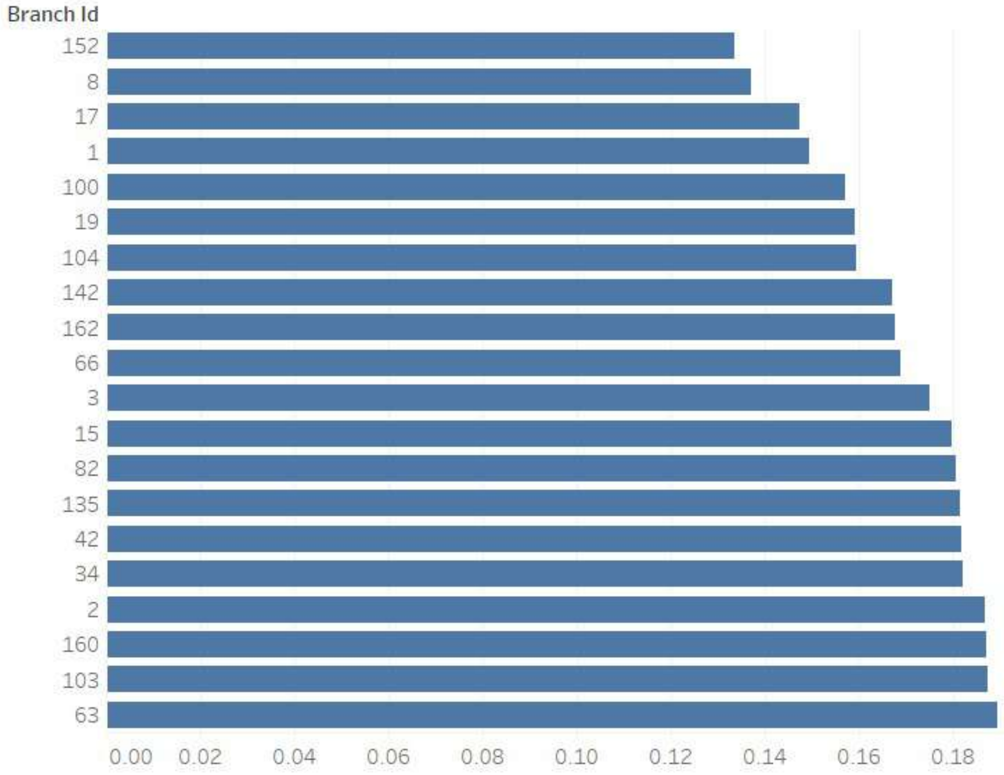
Bank Loan Defaulter Data Analysis

2/2

20 Branches having maximum proportion of defaulters



20 Branches having minimum proportion of defaulters



This information is suitable to identify branches with maximum proportion of defaulters and identify possibility of unfair practices in these branches.
Similarly, branches with minimum proportion of defaulters can be identified and employees in these branches can be rewarded appropriately.