Loantap Logistic Regression Business Casestudy

1. Data Preprocessing

- Import Required Libraries

In [ ]:
```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

- Downloading Dataset

In [ ]:
```
url = 'https://drive.google.com/file/d/1ZPYj7CZCfxntE8p2Lze_4QO4MyEOy6_d/view?usp=sharing'
path = 'https://drive.google.com/uc?export=download&id='+url.split('/')[-2]
df = pd.read_csv(path)
```

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

Out[ ]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	verification_status
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	Not Verified
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	Not Verified
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	Source Verified
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 years	RENT	54000.0	Not Verified
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years	MORTGAGE	55000.0	Verified

- Basic Data Inspection

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

Out[ ]: (396030, 27)

Data dictionary:

1. ***loan_amnt*** : The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
2. ***term*** : The number of payments on the loan. Values are in months and can be either 36 or 60. (EMI Months)
3. ***int_rate*** : Interest Rate on the loan some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
4. ***installment*** : The monthly payment owed by the borrower if the loan originates. (Monthly EMI)
5. ***grade*** : LoanTap assigned loan grade (A,B,C,D,E,F,G)
6. ***sub_grade*** : LoanTap assigned loan subgrade
7. ***emp_title*** :The job title supplied by the Borrower when applying for the loan.
8. ***emp_length*** : Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years. (Experience)
9. ***home_ownership*** : The home ownership status provided by the borrower during registration or obtained from the credit report. (Own / Rent / Mortage etc)
10. ***annual_inc*** : The self-reported annual income provided by the borrower during registration. (Salary)
11. ***verification_status*** : Indicates if income was verified by LoanTap, not verified, or if the income source was verified
12. ***issue_d*** : The month which the loan was funded (Loan Starting Date)
13. ***loan_status*** : Current status of the loan - Target Variable (Fully Paid or Charged off)

- Charged off Means borrower might not able to pay full amount (like Farmer Loans etc)

14. ***purpose*** : A category provided by the borrower for the loan request.

15. ***title*** : The loan title provided by the borrower
16. ***dti*** : A ratio calculated using the borrower’s total monthly debt payments on the total debt obligations, excluding mortgage and the requested LoanTap loan, divided by the borrower’s self-reported monthly income.
17. ***earliest_cr_line*** :The month the borrower's earliest reported credit line was opened
18. ***open_acc*** : The number of open credit lines in the borrower's credit file.
19. ***pub_rec*** : Number of derogatory public records (How many payments missed, bankruptcy etc)
20. ***revol_bal*** : Total credit revolving balance
21. ***revol_util*** : Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
22. ***total_acc*** : The total number of credit lines currently in the borrower's credit file
23. ***initial_list_status*** : The initial listing status of the loan. Possible values are – W, F
24. ***application_type*** : Indicates whether the loan is an individual application or a joint application with two co-borrowers
25. ***mort_acc*** : Number of mortgage accounts.
26. ***pub_rec_bankruptcies*** : Number of public record bankruptcies
27. ***Address***: Address of the individual

In [ ]:
```
np.round(df.isnull().sum() / df.shape[0],2) * 100
```

Out[ ]:
```
loan_amnt               0.0
term                    0.0
int_rate                0.0
installment             0.0
grade                   0.0
sub_grade               0.0
emp_title               6.0
emp_length              5.0
home_ownership          0.0
annual_inc              0.0
verification_status     0.0
issue_d                 0.0
loan_status             0.0
purpose                 0.0
title                   0.0
dti                     0.0
earliest_cr_line        0.0
open_acc                0.0
pub_rec                 0.0
revol_bal               0.0
revol_util              0.0
total_acc               0.0
initial_list_status     0.0
application_type        0.0
mort_acc               10.0
pub_rec_bankruptcies    0.0
address                 0.0
dtype: float64
```

- Notes
  - 10% of Null Values present in Mort_acc
  - emp_title,emp_length has 5%,6% Null Values Respectively

In [ ]:
```
df.duplicated().sum()
```

Out[ ]:
```
0
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   loan_amnt             396030 non-null  float64
 1   term                  396030 non-null  object
 2   int_rate              396030 non-null  float64
 3   installment           396030 non-null  float64
 4   grade                 396030 non-null  object
 5   sub_grade             396030 non-null  object
 6   emp_title             373103 non-null  object
 7   emp_length            377729 non-null  object
 8   home_ownership        396030 non-null  object
 9   annual_inc            396030 non-null  float64
 10  verification_status   396030 non-null  object
 11  issue_d               396030 non-null  object
 12  loan_status           396030 non-null  object
 13  purpose               396030 non-null  object
 14  title                 394274 non-null  object
 15  dti                   396030 non-null  float64
 16  earliest_cr_line      396030 non-null  object
 17  open_acc              396030 non-null  float64
 18  pub_rec               396030 non-null  float64
 19  revol_bal             396030 non-null  float64
 20  revol_util            395754 non-null  float64
 21  total_acc             396030 non-null  float64
 22  initial_list_status   396030 non-null  object
 23  application_type      396030 non-null  object
 24  mort_acc              358235 non-null  float64
 25  pub_rec_bankruptcies  395495 non-null  float64
 26  address               396030 non-null  object
dtypes: float64(12), object(15)
memory usage: 81.6+ MB
```

In [ ]:
```
df.describe()
```

Out[ ]:

	loan_amnt	int_rate	installment	annual_inc	dti	open_acc	pub_rec	revol_bal	revol
count	396030.000000	396030.000000	396030.000000	3.960300e+05	396030.000000	396030.000000	396030.000000	3.960300e+05	395754.000
mean	14113.888089	13.639400	431.849698	7.420318e+04	17.379514	11.311153	0.178191	1.584454e+04	53.791
std	8357.441341	4.472157	250.727790	6.163762e+04	18.019092	5.137649	0.530671	2.059184e+04	24.452
min	500.000000	5.320000	16.080000	0.000000e+00	0.000000	0.000000	0.000000	0.000000e+00	0.000
25%	8000.000000	10.490000	250.330000	4.500000e+04	11.280000	8.000000	0.000000	6.025000e+03	35.800
50%	12000.000000	13.330000	375.430000	6.400000e+04	16.910000	10.000000	0.000000	1.118100e+04	54.800
75%	20000.000000	16.490000	567.300000	9.000000e+04	22.980000	14.000000	0.000000	1.962000e+04	72.900
max	40000.000000	30.990000	1533.810000	8.706582e+06	9999.000000	90.000000	86.000000	1.743266e+06	892.300

In [ ]:

```
df.dtypes
```

Out[ ]:

```
loan_amnt               float64
term                     object
int_rate                float64
installment             float64
grade                    object
sub_grade                object
emp_title                object
emp_length               object
home_ownership           object
annual_inc              float64
verification_status      object
issue_d                  object
loan_status              object
purpose                  object
title                    object
dti                     float64
earliest_cr_line         object
open_acc                float64
pub_rec                 float64
revol_bal               float64
revol_util              float64
total_acc               float64
initial_list_status      object
application_type         object
mort_acc                float64
pub_rec_bankruptcies    float64
address                  object
dtype: object
```

In [ ]:

```
df.describe(include='object')
```

Out[ ]:

	term	grade	sub_grade	emp_title	emp_length	home_ownership	verification_status	issue_d	loan_status	purpose
count	396030	396030	396030	373103	377729	396030	396030	396030	396030	396030
unique	2	7	35	173105	11	6	3	115	2	14
top	36 months	B	B3	Teacher	10+ years	MORTGAGE	Verified	Oct-2014	Fully Paid	debt_consolidation
freq	302005	116018	26655	4389	126041	198348	139563	14846	318357	234507

- Address column expanding

In [ ]:
```
df['address']
```

Out[ ]:
```
0            0174 Michelle Gateway\r\nMendozaberg, OK 22690
1         1076 Carney Fort Apt. 347\r\nLoganmouth, SD 05113
2         87025 Mark Dale Apt. 269\r\nNew Sabrina, WV 05113
3                   823 Reid Ford\r\nDelacruzside, MA 00813
4                    679 Luna Roads\r\nGreggshire, VA 11650
                                ...
396025     12951 Williams Crossing\r\nJohnnyville, DC 30723
396026    0114 Fowler Field Suite 028\r\nRachelborough, ...
396027    953 Matthew Points Suite 414\r\nReedfort, NY 7...
396028    7843 Blake Freeway Apt. 229\r\nNew Michael, FL...
396029        787 Michelle Causeway\r\nBriannaton, AR 48052
Name: address, Length: 396030, dtype: object
```

In [ ]:
```
df['state'] = df['address'].str.split(', ').str[1].str.split(' ').str[0]

"""
Explanation :
  1. df['address'].str.split(', ') it will divide address into 2 parts
  2. we want state part so we are going to check 2nd part i,.e 1 then in the second part Zero index is State
"""
```

Out[ ]: "\nExplanation :\n 1. df['address'].str.split(', ') it will divide address into 2 parts\n 2. we want state part so we are g
oing to check 2nd part i,.e 1 then in the second part Zero index is State\n"

In [ ]:
```
pd.set_option('display.max_columns', 28)
df.head()
```

Out[ ]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	verification_status
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	Not Verified
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	Not Verified
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	Source Verified
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 years	RENT	54000.0	Not Verified
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years	MORTGAGE	55000.0	Verified

In [ ]:
```
# Dropping off address columns
df.drop('address',axis=1,inplace=True) # Column so axis 1
```

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

Out[ ]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	verification_status
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	Not Verified
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	Not Verified
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	Source Verified
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 years	RENT	54000.0	Not Verified
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years	MORTGAGE	55000.0	Verified

In [ ]:

```
df['term'].unique()
```

Out[ ]:

```
array([' 36 months', ' 60 months'], dtype=object)
```

In [ ]:

```

```

- Column Segmentation

In [ ]:

```
cat_col = [i for i in df.columns if df[i].dtype == 'object']
num_col = [i for i in df.columns if df[i].dtype != 'object']
```

In [ ]:

```
num_col
```

Out[ ]:
```
['loan_amnt',
 'int_rate',
 'installment',
 'annual_inc',
 'dti',
 'open_acc',
 'pub_rec',
 'revol_bal',
 'revol_util',
 'total_acc',
 'mort_acc',
 'pub_rec_bankruptcies']
```

In [ ]:
```
cat_col
```

Out[ ]:
```
['term',
 'grade',
 'sub_grade',
 'emp_title',
 'emp_length',
 'home_ownership',
 'verification_status',
 'issue_d',
 'loan_status',
 'purpose',
 'title',
 'earliest_cr_line',
 'initial_list_status',
 'application_type',
 'state']
```

- Outliers Detection

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

Out[ ]:
```
(396030, 27)
```

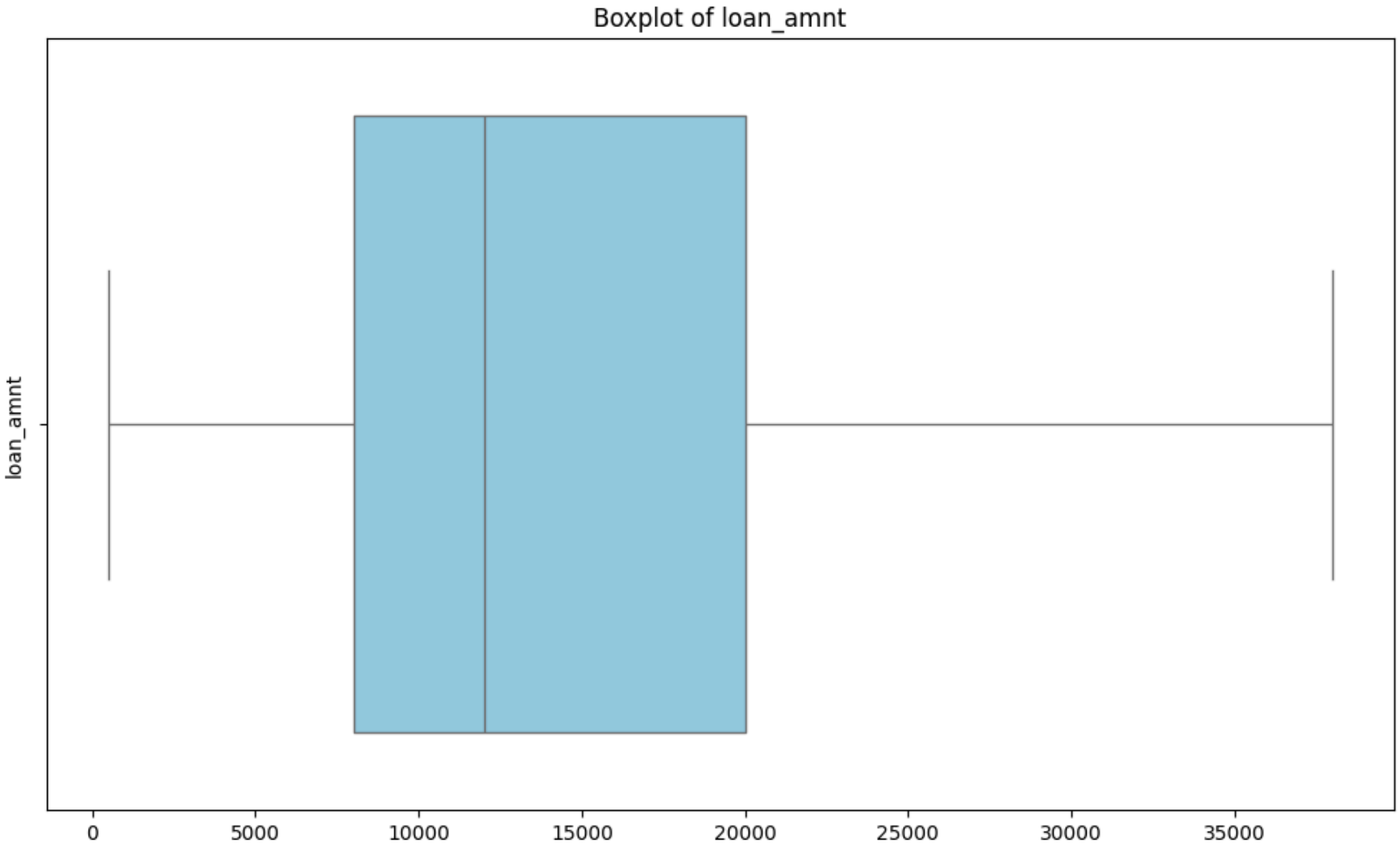
In [ ]:
```
def remove_outliers_iqr(series):
    Q1 = series.quantile(0.25)
```

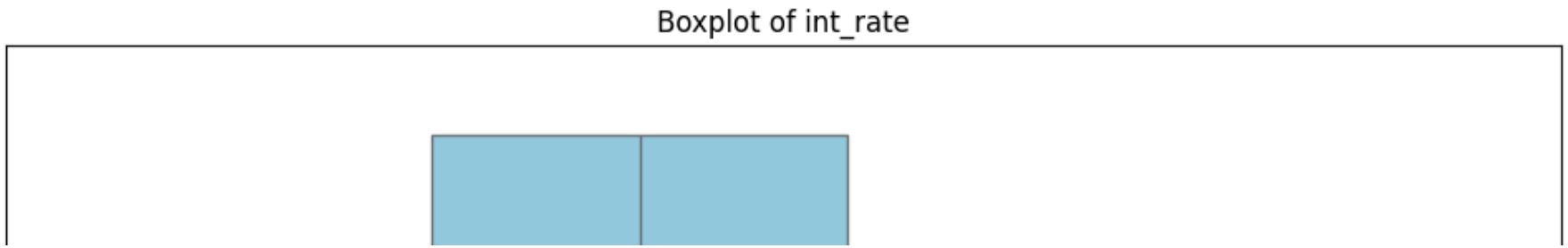
```
    Q3 = series.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return series[(series >= lower_bound) & (series <= upper_bound)]

for col in num_col:
    df[col] = remove_outliers_iqr(df[col])

df.reset_index(drop=True, inplace=True)
```

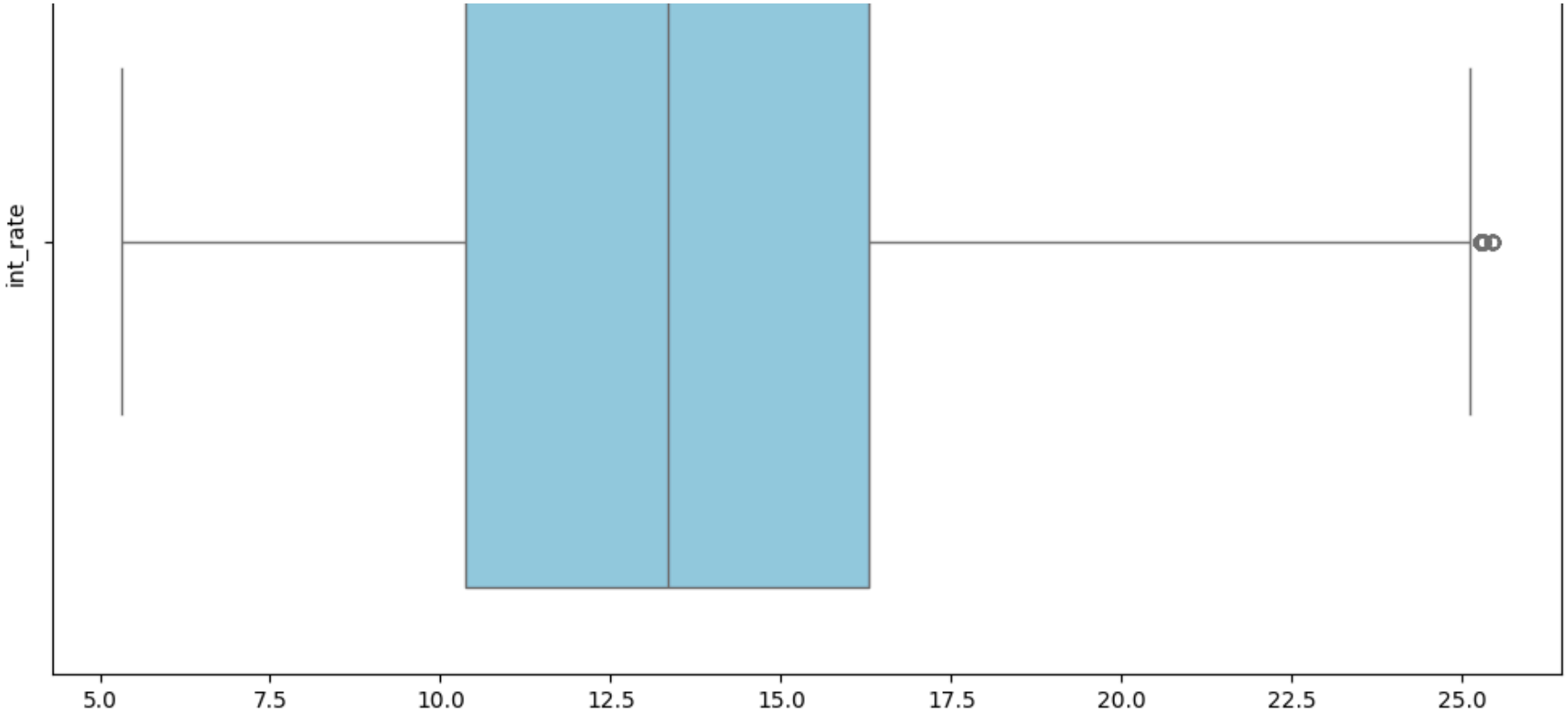
In [ ]:
```
df.shape
```

Out[ ]: (396030, 27)

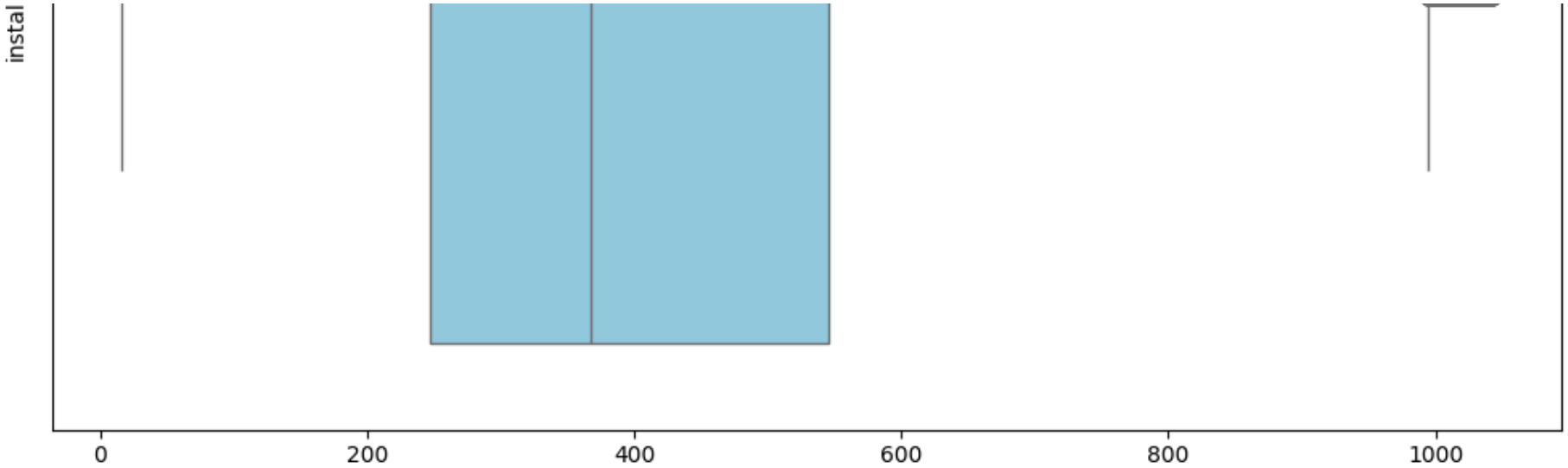
In [ ]:
```
fig, axes = plt.subplots(len(df[num_col].columns), 1, figsize=(10, 6 * len(df[num_col].columns)))

for i, col in enumerate(df[num_col].columns):
    sns.boxplot(x=df[num_col][col], ax=axes[i], color='skyblue')
    axes[i].set_title(f'Boxplot of {col}')
    axes[i].set_xlabel('')
    axes[i].set_ylabel(col)

plt.tight_layout()
plt.show()
```


Boxplot of loan_amnt
loan_amnt
0
5000
10000
15000
20000
25000
30000
35000

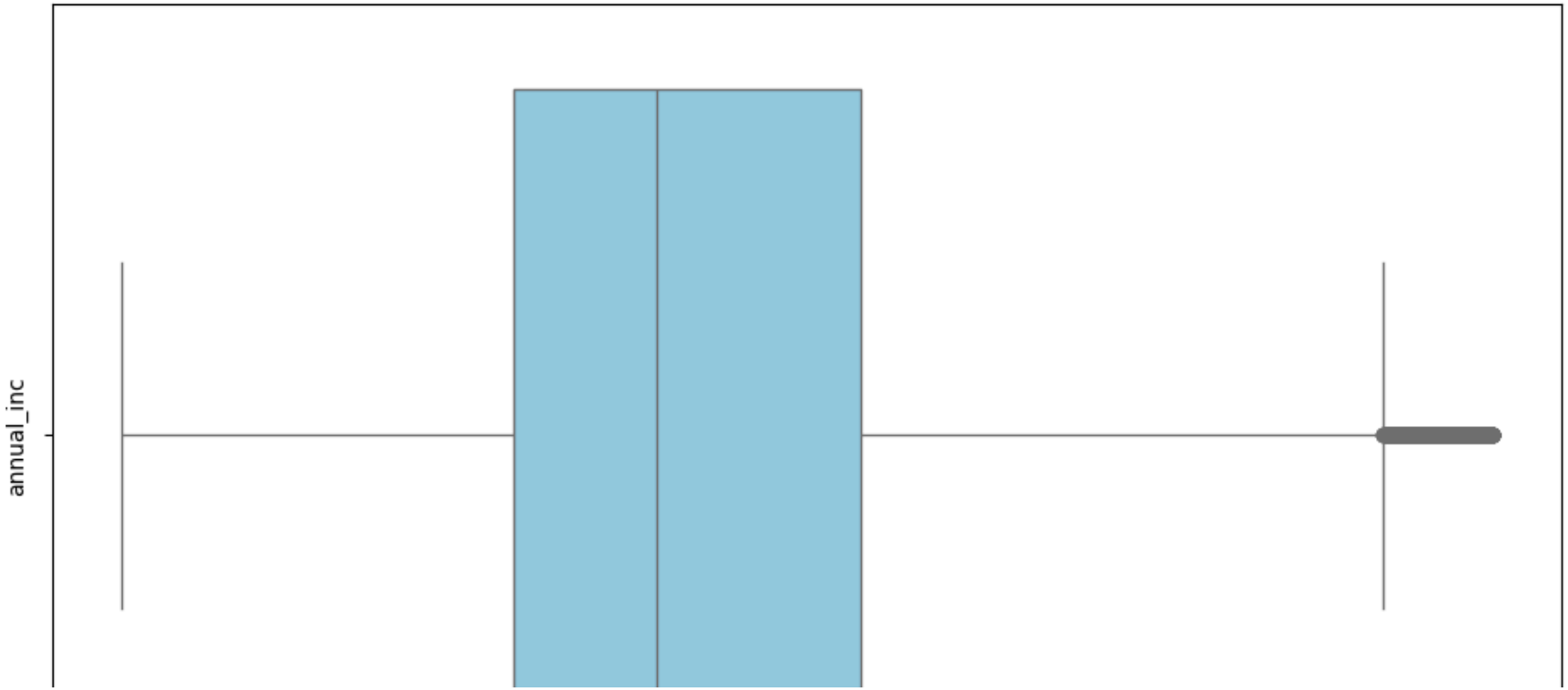

Boxplot of int_rate

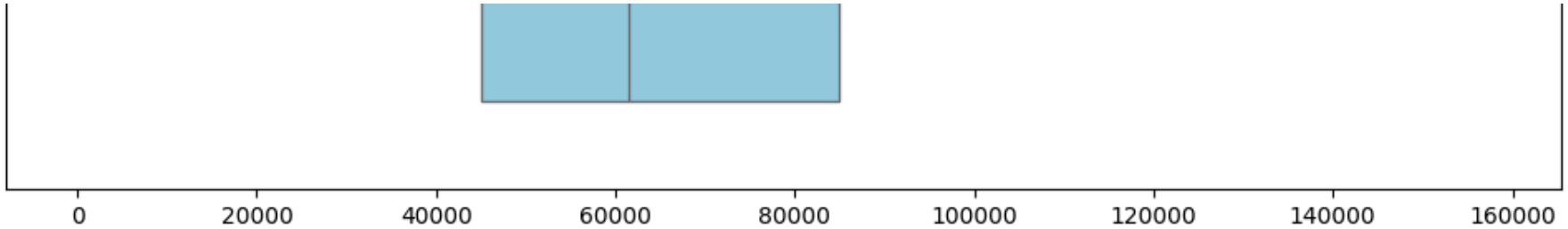


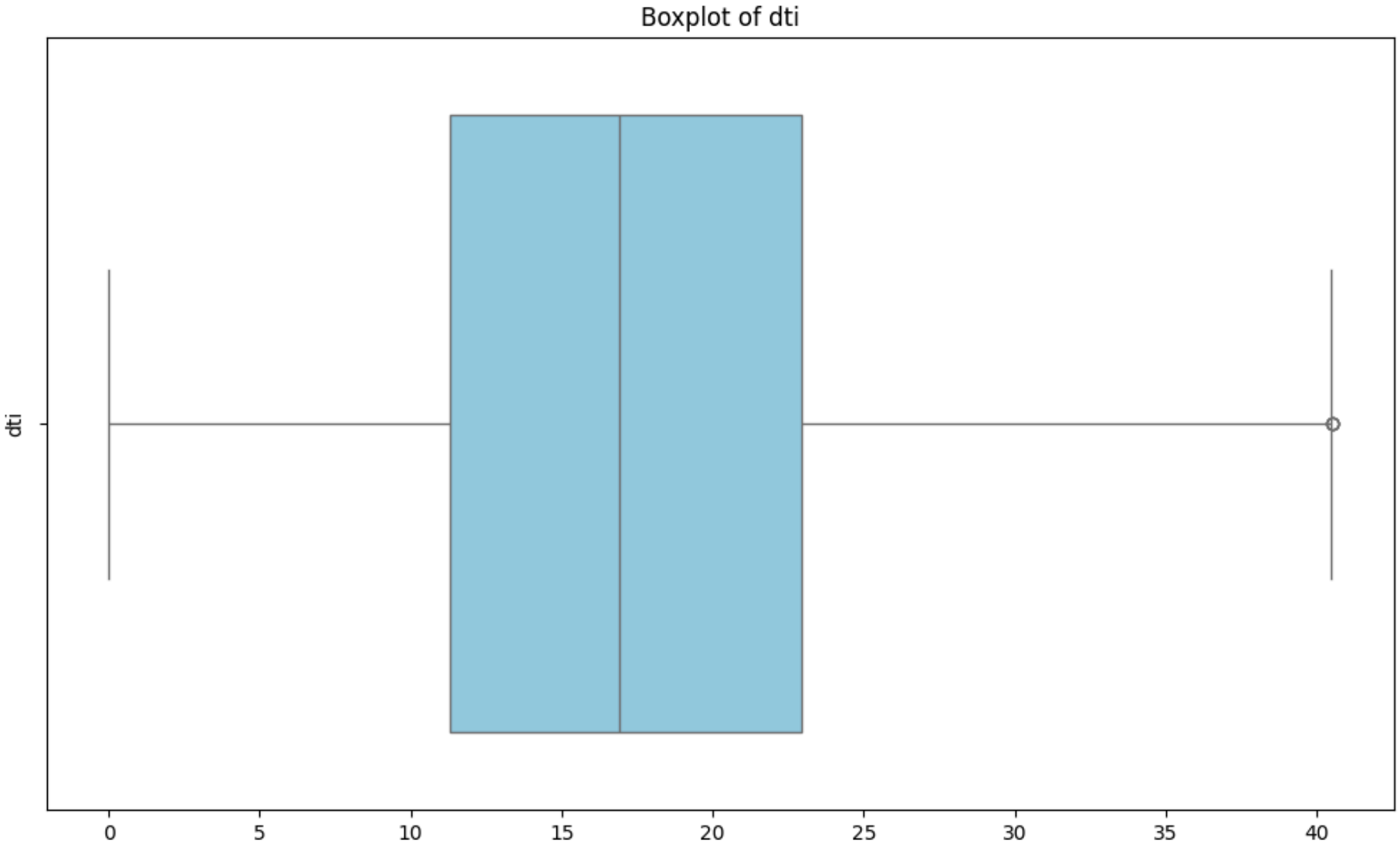
Boxplot of installment

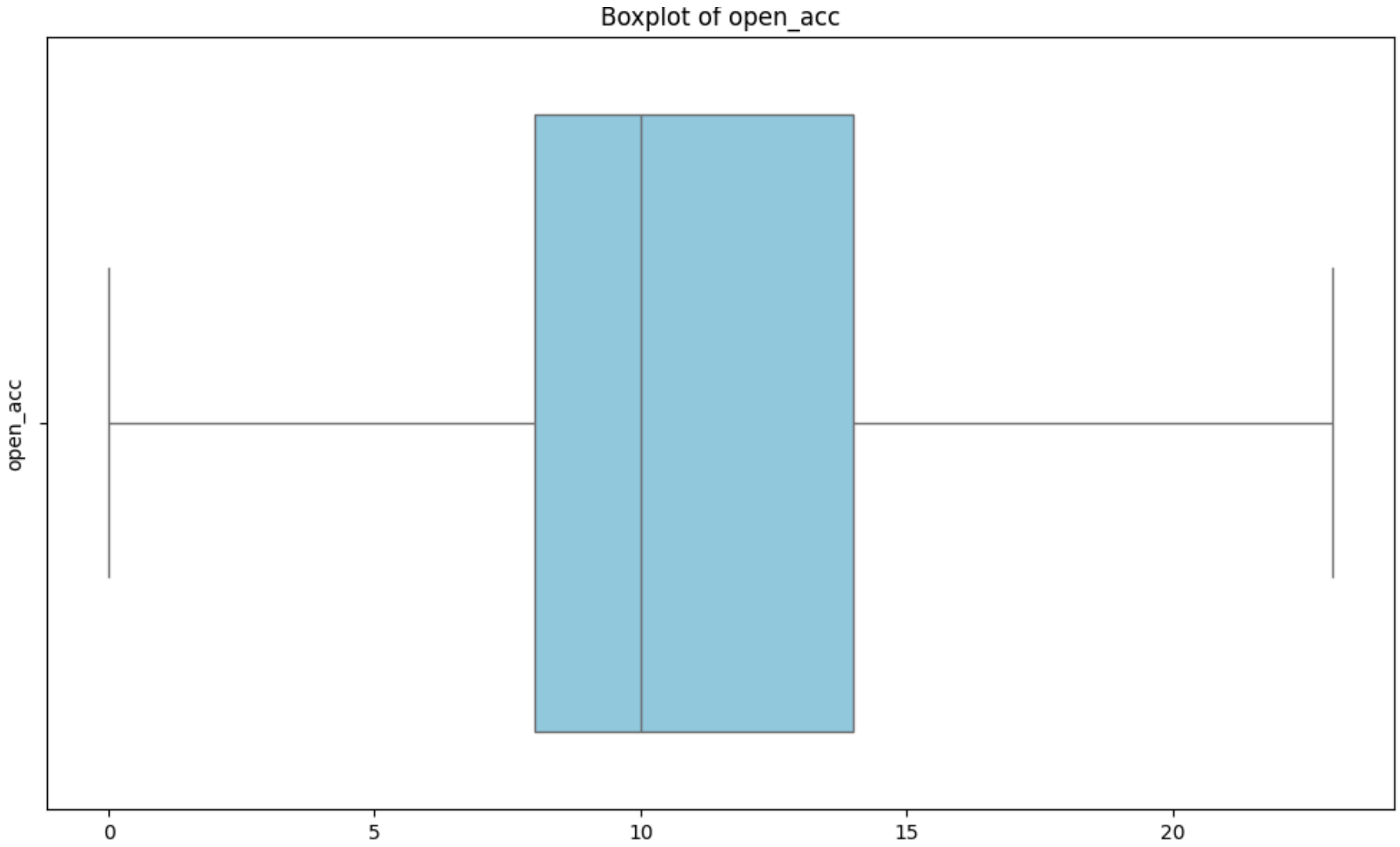


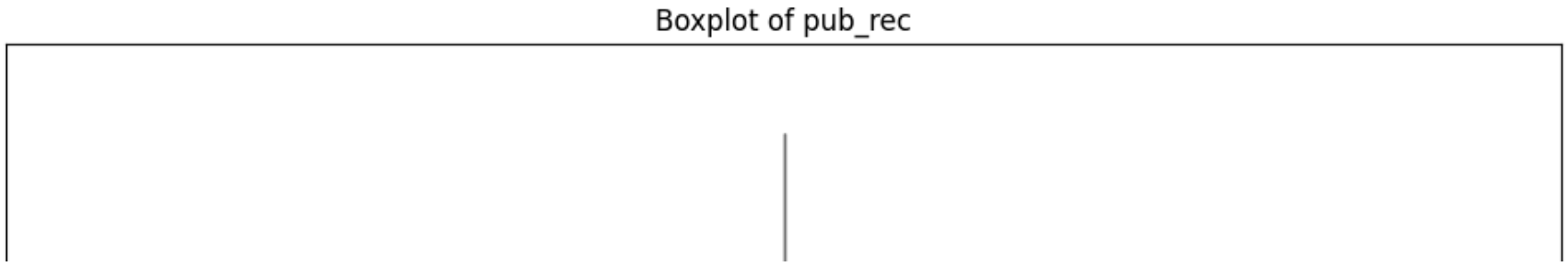
Boxplot of annual_inc

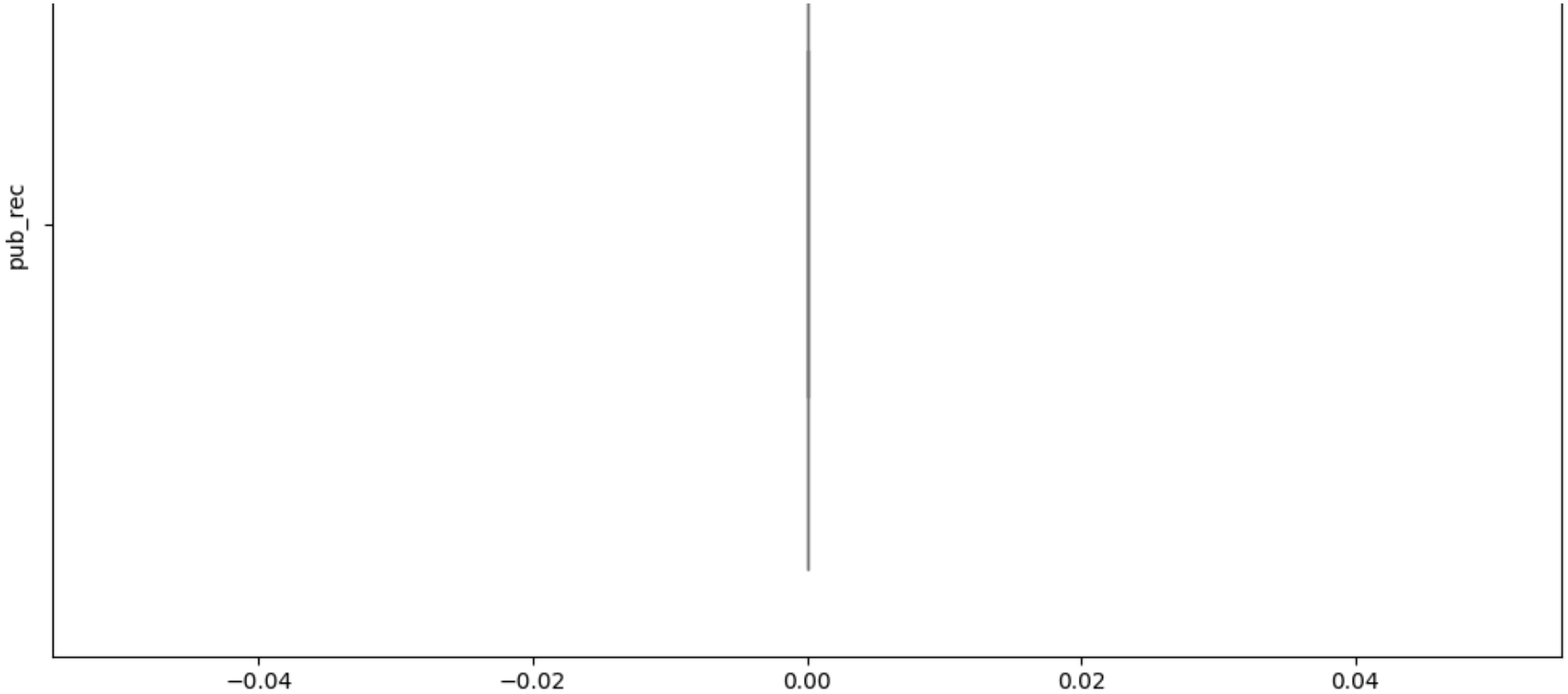


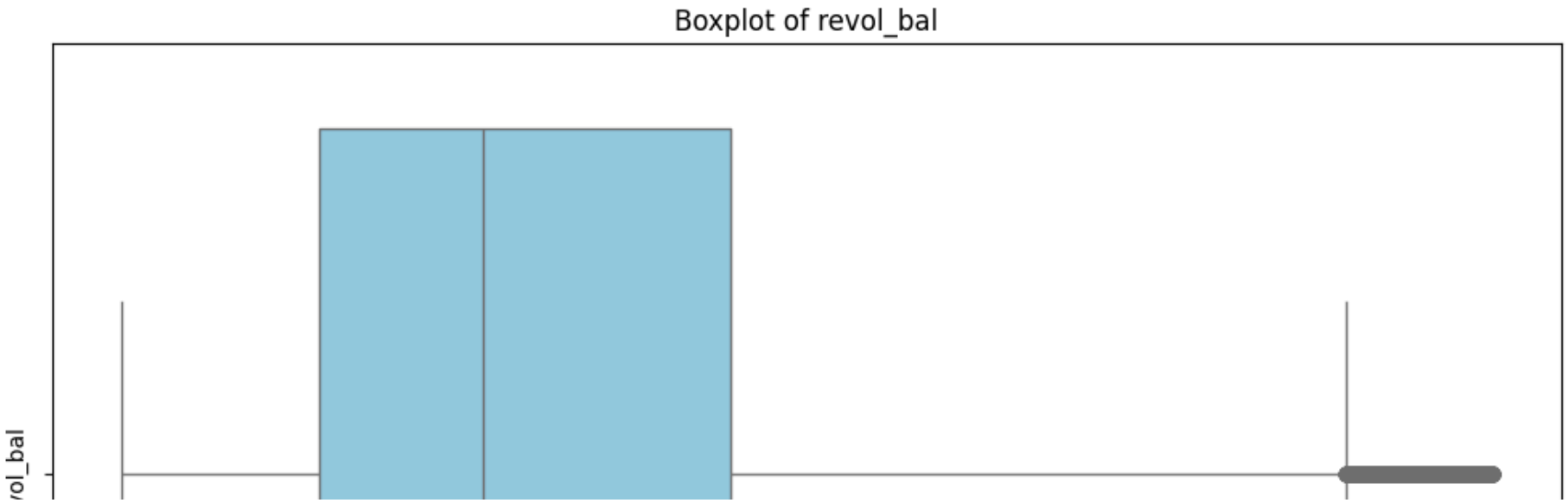

0
20000
40000
60000
80000
100000
120000
140000
160000

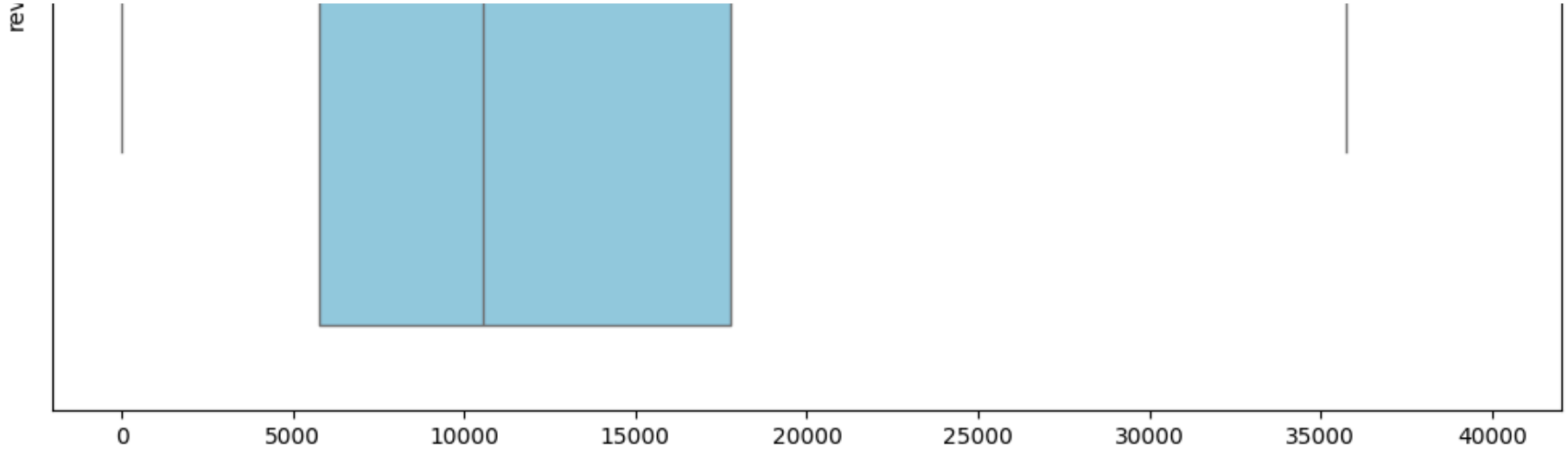

Boxplot of dti
dti
0
5
10
15
20
25
30
35
40


Boxplot of open_acc
open_acc
0
5
10
15
20

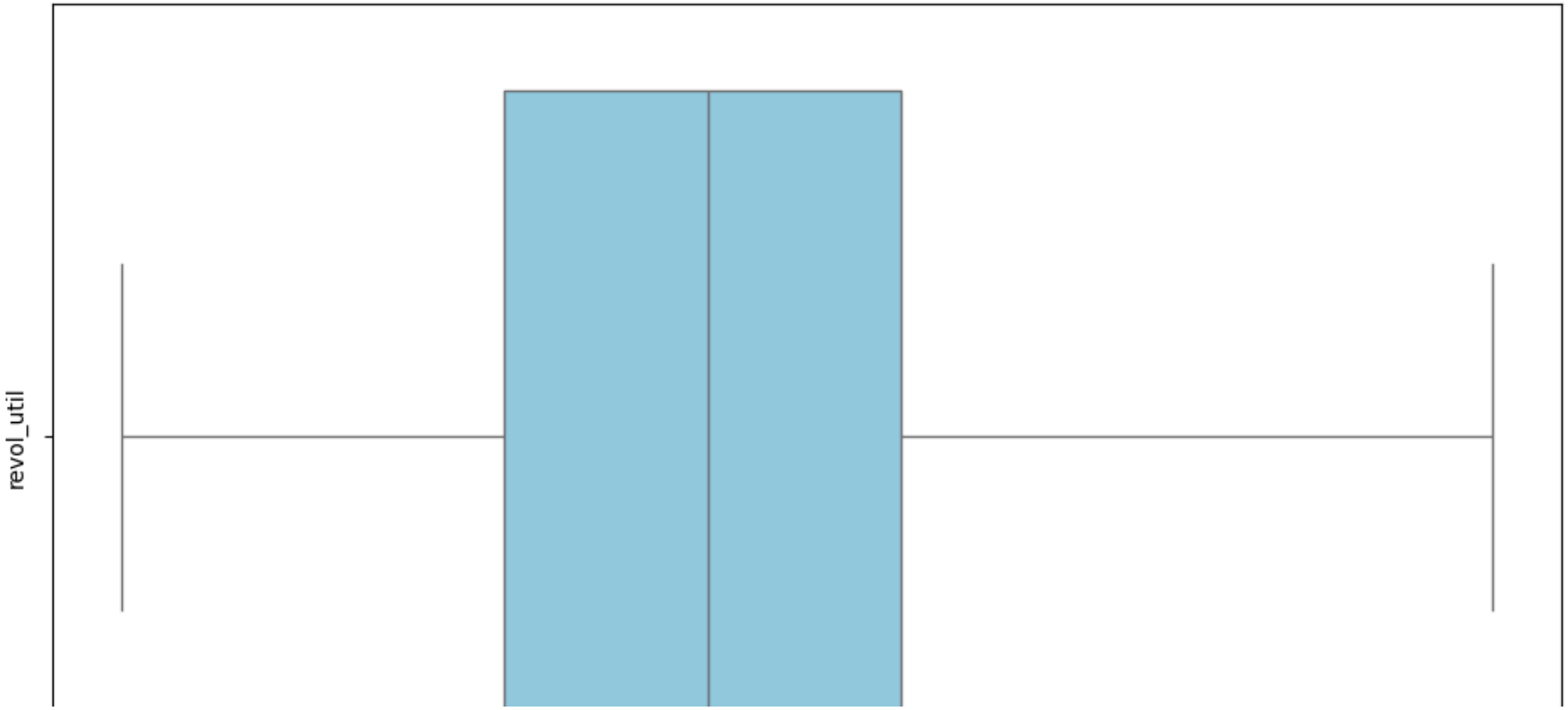

Boxplot of pub_rec

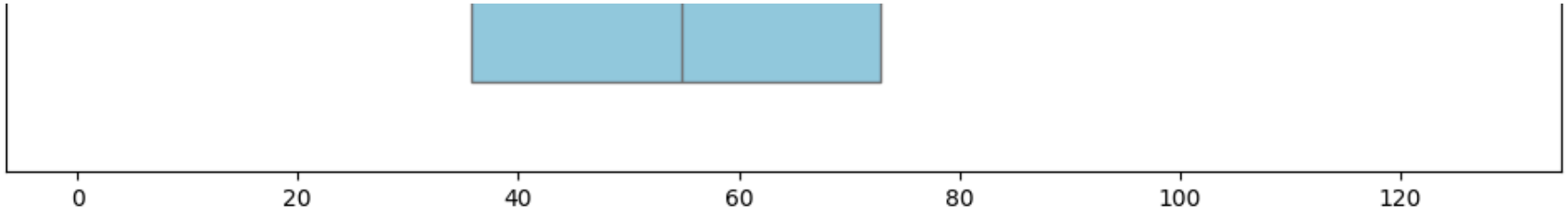

pub_rec
−0.04
−0.02
0.00
0.02
0.04


Boxplot of revol_bal
vol_bal

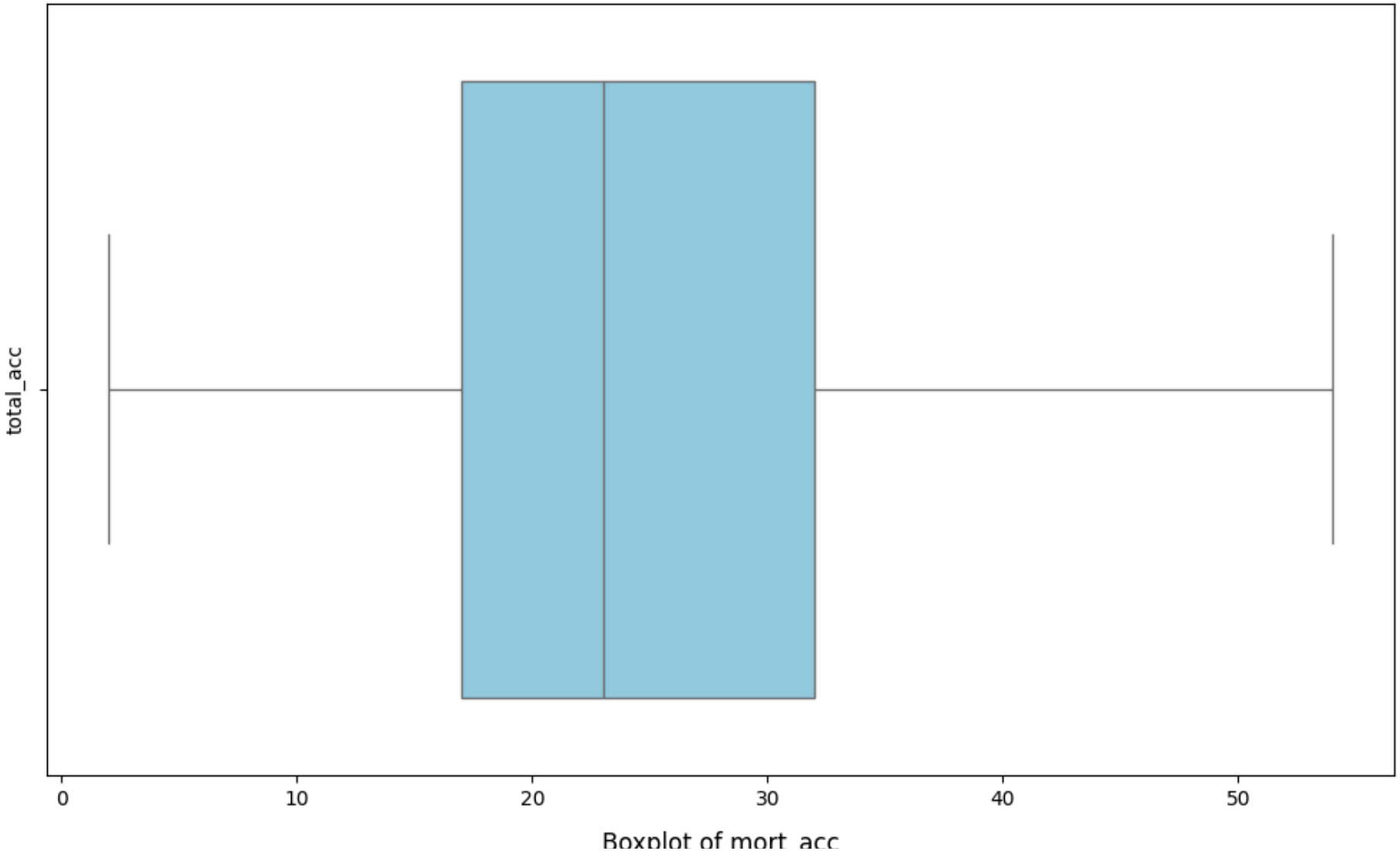


Boxplot of revol_util

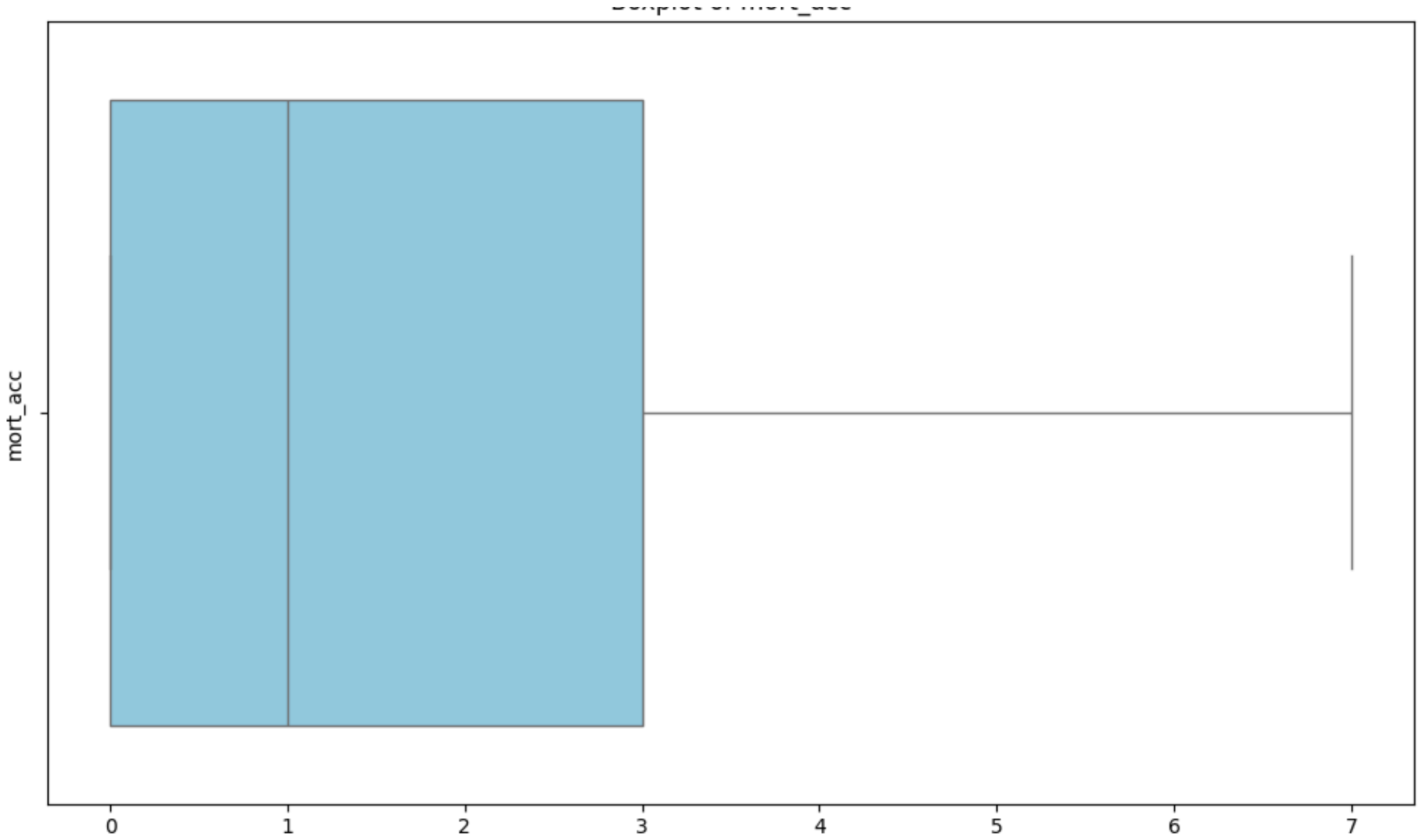




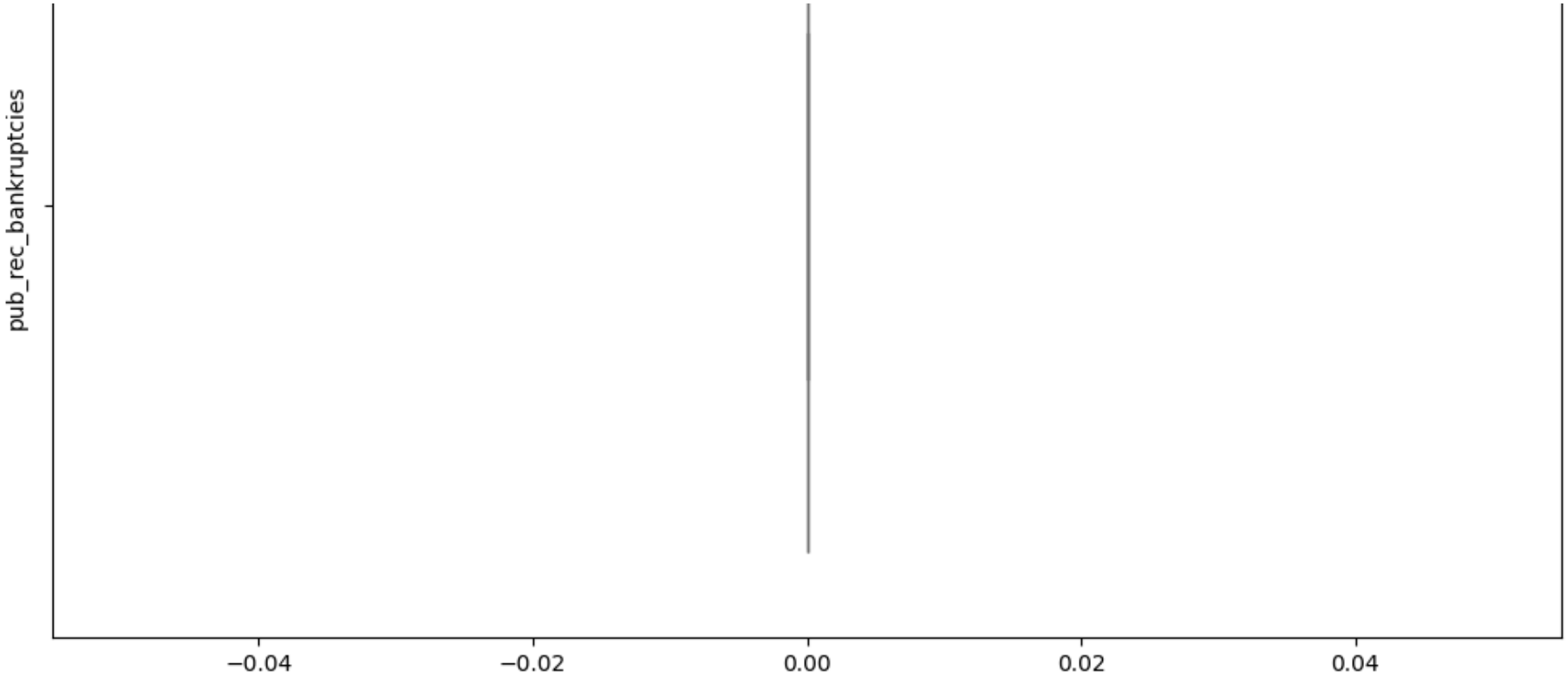
Boxplot of total_acc



Boxplot of mort_acc



Boxplot of pub_rec_bankruptcies



In [ ]:
```
num_col
```

Out[ ]:
```
['loan_amnt',
 'int_rate',
 'installment',
 'annual_inc',
 'dti',
 'open_acc',
 'pub_rec',
 'revol_bal',
 'revol_util',
 'total_acc',
 'mort_acc',
 'pub_rec_bankruptcies']
```

In [ ]:

In [ ]:
```
for i in cat_col:
    print(i,df[i].nunique())
    print(df[i].value_counts())
    print('='*50)
```

```
term 2
term
 36 months    302005
 60 months     94025
Name: count, dtype: int64
==================================================
grade 7
grade
B    116018
C    105987
A     64187
D     63524
E     31488
F     11772
G      3054
Name: count, dtype: int64
==================================================
sub_grade 35
sub_grade
B3    26655
B4    25601
C1    23662
C2    22580
B2    22495
B5    22085
C3    21221
C4    20280
B1    19182
A5    18526
C5    18244
D1    15993
A4    15789
D2    13951
D3    12223
D4    11657
A3    10576
A1     9729
D5     9700
A2     9567
E1     7917
E2     7431
```

```
E3     6207
E4     5361
E5     4572
F1     3536
F2     2766
F3     2286
F4     1787
F5     1397
G1     1058
G2      754
G3      552
G4      374
G5      316
Name: count, dtype: int64
==================================================
emp_title 173105
emp_title
Teacher                    4389
Manager                    4250
Registered Nurse           1856
RN                         1846
Supervisor                 1830
                           ...
Postman                       1
McCarthy & Holthus, LLC       1
jp flooring                   1
Histology Technologist        1
Gracon Services, Inc          1
Name: count, Length: 173105, dtype: int64
==================================================
emp_length 11
emp_length
10+ years    126041
2 years       35827
< 1 year      31725
3 years       31665
5 years       26495
1 year        25882
4 years       23952
6 years       20841
7 years       20819
```

```
8 years       19168
9 years       15314
Name: count, dtype: int64
==================================================
home_ownership 6
home_ownership
MORTGAGE    198348
RENT        159790
OWN          37746
OTHER          112
NONE            31
ANY              3
Name: count, dtype: int64
==================================================
verification_status 3
verification_status
Verified           139563
Source Verified    131385
Not Verified       125082
Name: count, dtype: int64
==================================================
issue_d 115
issue_d
Oct-2014    14846
Jul-2014    12609
Jan-2015    11705
Dec-2013    10618
Nov-2013    10496
            ...
Jul-2007       26
Sep-2008       25
Nov-2007       22
Sep-2007       15
Jun-2007        1
Name: count, Length: 115, dtype: int64
==================================================
loan_status 2
loan_status
Fully Paid     318357
Charged Off     77673
Name: count, dtype: int64
```

```
==================================================
purpose 14
purpose
debt_consolidation    234507
credit_card            83019
home_improvement       24030
other                  21185
major_purchase          8790
small_business          5701
car                     4697
medical                 4196
moving                  2854
vacation                2452
house                   2201
wedding                 1812
renewable_energy         329
educational              257
Name: count, dtype: int64
==================================================
title 48816
title
Debt consolidation            152472
Credit card refinancing        51487
Home improvement               15264
Other                          12930
Debt Consolidation             11608
                               ...
Graduation/Travel Expenses         1
Daughter's Wedding Bill            1
gotta move                         1
creditcardrefi                     1
Toxic Debt Payoff                  1
Name: count, Length: 48816, dtype: int64
==================================================
earliest_cr_line 684
earliest_cr_line
Oct-2000    3017
Aug-2000    2935
Oct-2001    2896
Aug-2001    2884
Nov-2000    2736
```

```
            ...
Jul-1958       1
Nov-1957       1
Jan-1953       1
Jul-1955       1
Aug-1959       1
Name: count, Length: 684, dtype: int64
==================================================
initial_list_status 2
initial_list_status
f    238066
w    157964
Name: count, dtype: int64
==================================================
application_type 3
application_type
INDIVIDUAL    395319
JOINT            425
DIRECT_PAY       286
Name: count, dtype: int64
==================================================
state 52
state
Box    14060
NJ      7091
WI      7081
LA      7068
NV      7038
AK      7034
VA      7022
MA      7022
VT      7005
NY      7004
MS      7003
TX      7000
SC      6973
ME      6972
AR      6969
OH      6969
GA      6967
ID      6958
```

```
IN      6958
KS      6945
WV      6944
RI      6940
MO      6939
IL      6934
WY      6933
NE      6927
HI      6927
IA      6926
FL      6921
AZ      6918
CO      6914
OK      6911
MN      6904
CT      6904
NC      6901
AL      6898
OR      6898
CA      6898
MD      6896
WA      6895
SD      6887
UT      6887
MT      6883
DE      6874
TN      6869
ND      6858
MI      6854
NM      6842
DC      6842
PA      6825
NH      6818
KY      6800
Name: count, dtype: int64
==================================================
```

In [ ]:
```
cat_col
```

Out[ ]:
```
['term',
 'grade',
 'sub_grade',
 'emp_title',
 'emp_length',
 'home_ownership',
 'verification_status',
 'issue_d',
 'loan_status',
 'purpose',
 'title',
 'earliest_cr_line',
 'initial_list_status',
 'application_type',
 'state']
```

- Handling Null Values

In [ ]:
```
np.round(df.isnull().sum() / df.shape[0],2) * 100
```

Out[ ]:
```
loan_amnt                0.0
term                     0.0
int_rate                 1.0
installment              3.0
grade                    0.0
sub_grade                0.0
emp_title                6.0
emp_length               5.0
home_ownership           0.0
annual_inc               4.0
verification_status      0.0
issue_d                  0.0
loan_status              0.0
purpose                  0.0
title                    0.0
dti                      0.0
earliest_cr_line         0.0
open_acc                 3.0
pub_rec                 15.0
revol_bal                5.0
revol_util               0.0
total_acc                2.0
initial_list_status      0.0
application_type         0.0
mort_acc                11.0
pub_rec_bankruptcies    12.0
state                    7.0
dtype: float64
```

- Employee Title

In [ ]:
```
df['emp_title'].isnull().sum()
```

Out[ ]:
```
22927
```

- There are 22K Records are having Null Values which 6% of total dataset size, since it is less than 10% I'm going with Imputation
- ***emp_title*** is categorical right we are using Mode as Imputation Technique

In [ ]:
```
df['emp_title'].fillna(df['emp_title'].mode()[0],inplace=True)
```

In [ ]:
```
# Cross check
df['emp_title'].isnull().sum()
```

Out[ ]:
```
0
```

- Employee Length

In [ ]:
```
df['emp_length'].isnull().sum()
```

Out[ ]:
```
18301
```

- There are 18K Records are having Null Values which 5% of total dataset size, since it is less than 10% I'm going with Imputation
- ***emp_length*** is categorical right we are using Mode as Imputation Technique

In [ ]:
```
df['emp_length'].fillna(df['emp_length'].mode()[0],inplace=True)
```

In [ ]:
```
df['emp_length'].isnull().sum()
```

Out[ ]:
```
0
```

- Mortrage Accounts

In [ ]:
```
df['mort_acc'].isnull().sum()
```

Out[ ]:
```
44638
```

- There are 37K Records are having Null Values which 10% of total dataset size, since it is less than 10% I'm going with Imputation
- ***mort_acc*** is categorical right we are using Mode as Imputation Technique

In [ ]:
```
df['mort_acc'].fillna(df['mort_acc'].mode()[0],inplace=True)
```

In [ ]:
```
df['mort_acc'].isnull().sum()
```

Out[ ]: 0

- State

In [ ]:
```
df['state'].isnull().sum()
```

Out[ ]: 28324

- State Having 7% Null Values

In [ ]:
```
df['state'].fillna(df['state'].mode()[0],inplace=True)
```

In [ ]:
```
df['state'].isnull().sum()
```

Out[ ]: 0

In [ ]:
```
np.round(df.isnull().sum() / df.shape[0],2) * 100
```

Out[ ]:
```
loan_amnt               0.0
term                    0.0
int_rate                1.0
installment             3.0
grade                   0.0
sub_grade               0.0
emp_title               0.0
emp_length              0.0
home_ownership          0.0
annual_inc              4.0
verification_status     0.0
issue_d                 0.0
loan_status             0.0
purpose                 0.0
title                   0.0
dti                     0.0
earliest_cr_line        0.0
open_acc                3.0
pub_rec                15.0
revol_bal               5.0
revol_util              0.0
total_acc               2.0
initial_list_status     0.0
application_type        0.0
mort_acc                0.0
pub_rec_bankruptcies   12.0
state                   0.0
dtype: float64
```

- No Null Values

2. Exploratory Data Analysis

- Disclaimer : I used plotly for graphs for my practice

In [ ]:
```
df.describe()
```

Out[ ]:

	loan_amnt	int_rate	installment	annual_inc	dti	open_acc	pub_rec	revol_bal	revol_util
count	395839.000000	392253.000000	384780.000000	379330.000000	395755.000000	385723.000000	338272.0	374771.000000	395742.000000
mean	14101.427032	13.516850	410.221050	67180.888026	17.328272	10.873303	0.0	12702.161173	53.787172
std	8340.175521	4.313596	219.215703	29761.853088	8.094345	4.383015	0.0	9008.820738	24.411716
min	500.000000	5.320000	16.080000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000
25%	8000.000000	10.360000	246.655000	45000.000000	11.280000	8.000000	0.0	5778.000000	35.800000
50%	12000.000000	13.330000	367.375000	61512.500000	16.900000	10.000000	0.0	10553.000000	54.800000
75%	20000.000000	16.290000	545.670000	85000.000000	22.965000	14.000000	0.0	17768.000000	72.900000
max	38000.000000	25.440000	1042.730000	157500.000000	40.520000	23.000000	0.0	40012.000000	128.100000

In [ ]:

```
pip install plotly==5.22.0
```

```
Requirement already satisfied: plotly==5.22.0 in /usr/local/lib/python3.10/dist-packages (5.22.0)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly==5.22.0) (8.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from plotly==5.22.0) (24.0)
```

- ***Loan Amount Distribution***

In [ ]:

```
import plotly.express as px
fig = px.histogram(df, x=df['loan_amnt'], nbins=len(set(df)),color_discrete_sequence=['green'])
# Update layout
fig.update_layout(
    title='Loan Amount Distribution',
    xaxis_title="Value",
    yaxis_title="Frequency"
)
```

In [ ]:
```
((33.959 + 39.21 + 37.93 + 46.20 + 36.13 + 36.87)*1000 / df.shape[0])* 100
```

Out[ ]:
```
58.15190768376134
```

- Note
  - 58% of People Take loan in between 4K - 16K

- ***Interest Rate***

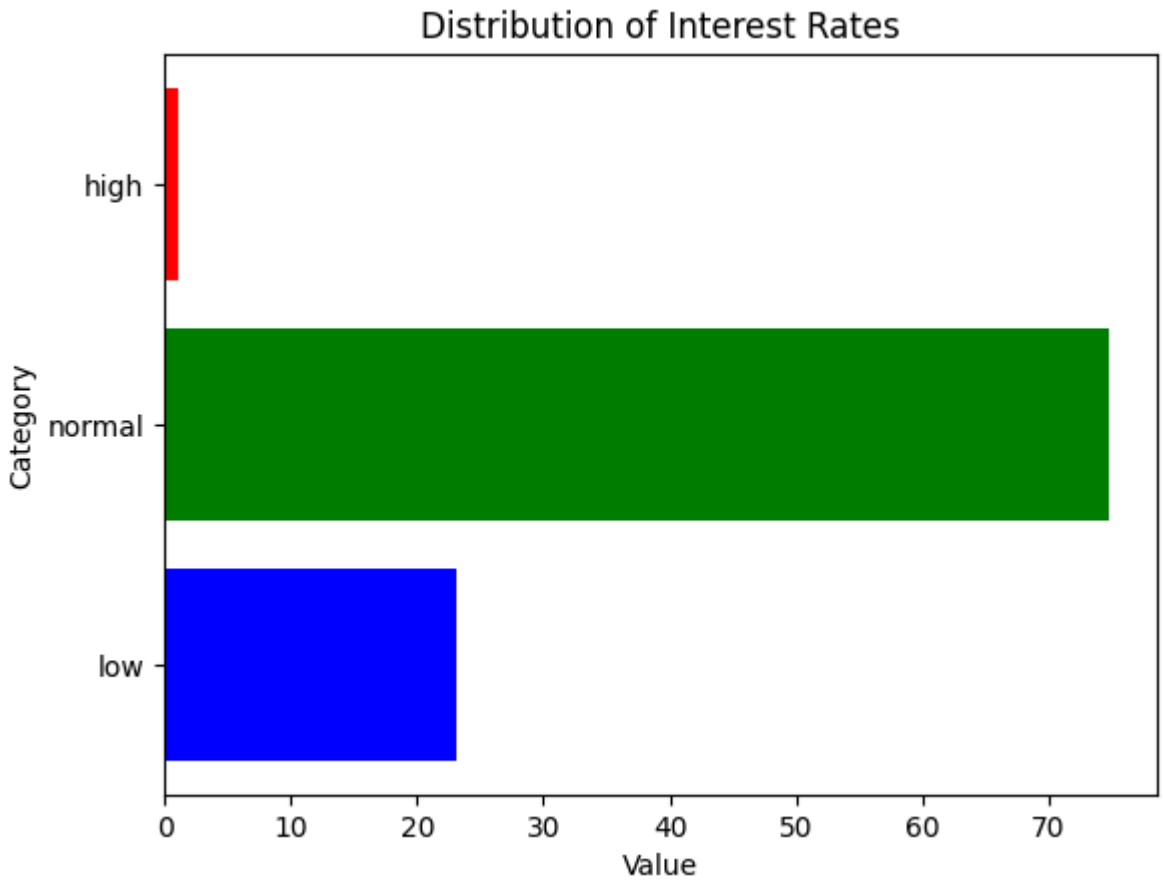
In [ ]:
```
fig = px.histogram(df, x=df['int_rate'], nbins=len(set(df)),color_discrete_sequence=['blue'])
# Update layout
fig.update_layout(
    title='Interest Rate Distribution',
    xaxis_title="Value",
    yaxis_title="Frequency"
)
```

In [ ]:
```
low = df[df['int_rate'] < 10 ].shape[0]/df.shape[0] * 100
high = df[df['int_rate'] >24 ].shape[0]/df.shape[0] * 100
normal = df[(df['int_rate'] >= 10) & (df['int_rate'] <= 24)].shape[0]/df.shape[0] * 100
```

In [ ]:
```
low,normal,high
```

Out[ ]:
```
(23.133853495947278, 74.79610130545666, 1.1163295709920966)
```

In [ ]:
```
categories = ['low', 'normal', 'high']
values = [low,normal,high]
plt.barh(categories, values, color=['blue', 'green', 'red'])
plt.title('Distribution of Interest Rates')
plt.xlabel('Value')
plt.ylabel('Category')
plt.show()
```

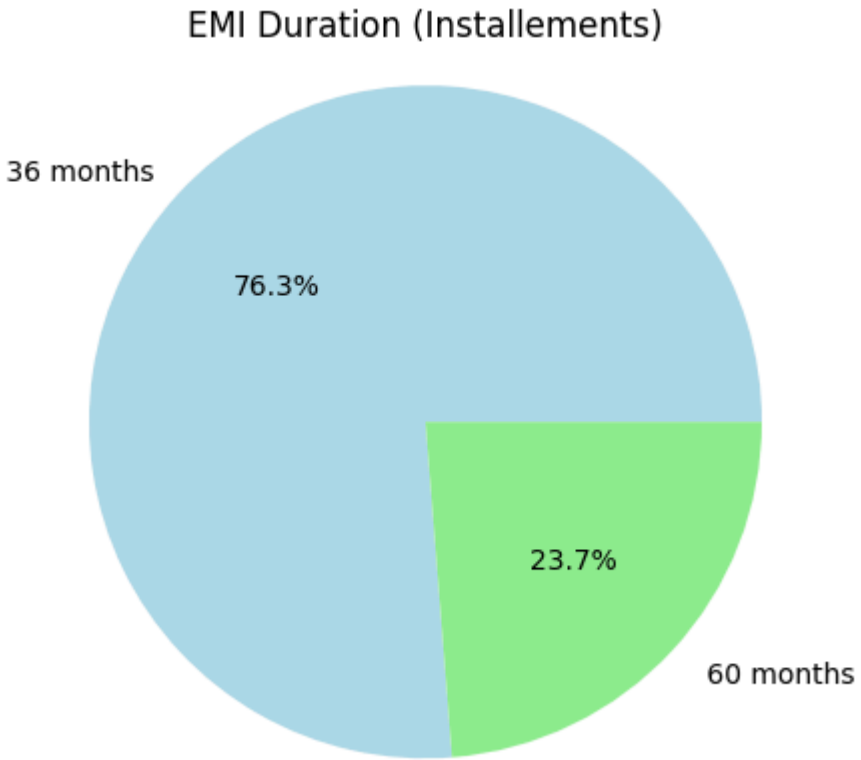


- Note
  - Interest Rate distribution is following Normal Distibution curve
  - 74% People have taken Medium Interest Rate (10-24) Rate

- Almost Quarter percent people taken Low Interest Rate (<10%)
- 2 percent People are Rich kids !

- ***EMI Time Distribution***

In [ ]:
```
term_counts = df['term'].value_counts(normalize=True) * 100
plt.pie(term_counts, labels=term_counts.index, autopct='%1.1f%%', colors=['lightblue', 'lightgreen'])
plt.title('EMI Duration (Installements)')
plt.axis('equal')
plt.show()
```



- Note

    - Most people prefer to take 36 Months (3 Years Installments)

- ***Installment per Month***

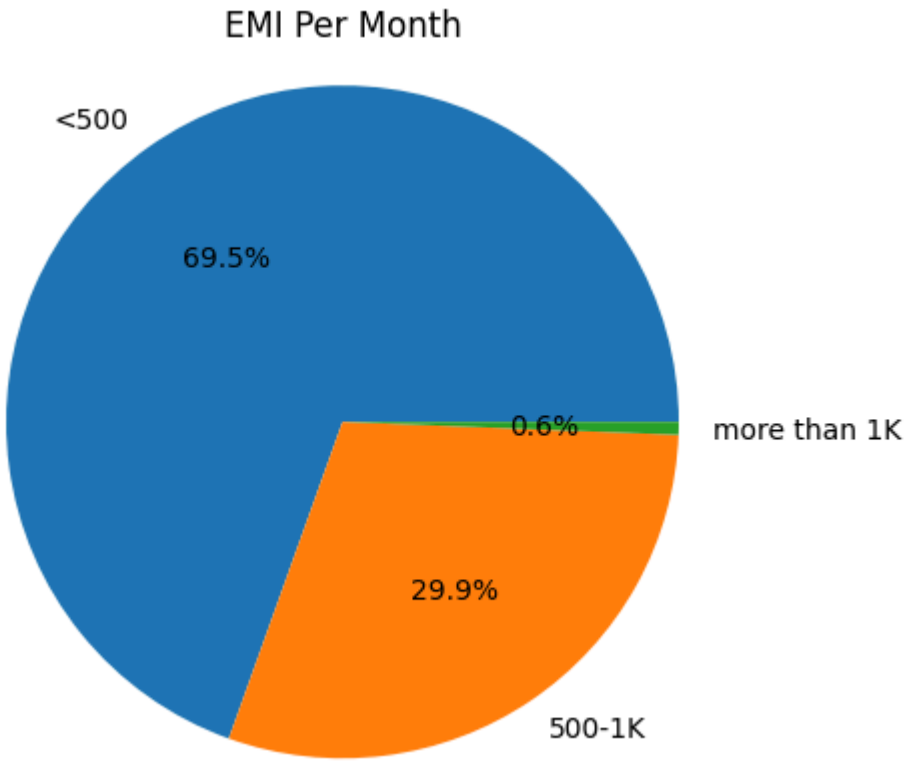
In [ ]:
```
fig = px.histogram(df, x=df['installment'], nbins=len(set(df)),color_discrete_sequence=['orange']) # using 27 Bins as same as
# Update layout
fig.update_layout(
    title='Installment per Month',
    xaxis_title="Value",
    yaxis_title="Frequency"
)
```

In [ ]:
```
low = df[df['installment'] < 500 ].shape[0]/df.shape[0] * 100
normal = df[(df['installment'] > 500) & (df['installment'] <= 1000)].shape[0]/df.shape[0] * 100
high = df[df['installment'] >1000 ].shape[0]/df.shape[0] * 100
```

In [ ]:
```
values
```

Out[ ]:
```
[23.133853495947278, 74.79610130545666, 1.1163295709920966]
```

In [ ]:
```
values = [low,normal,high]  # [67.53756028583693, 29.027598919273796, 3.433325758149635]
labels = ['<500', '500-1K', 'more than 1K']
plt.pie(values, labels=labels, autopct='%1.1f%%')
plt.title('EMI Per Month')
plt.axis('equal')
plt.show()
```



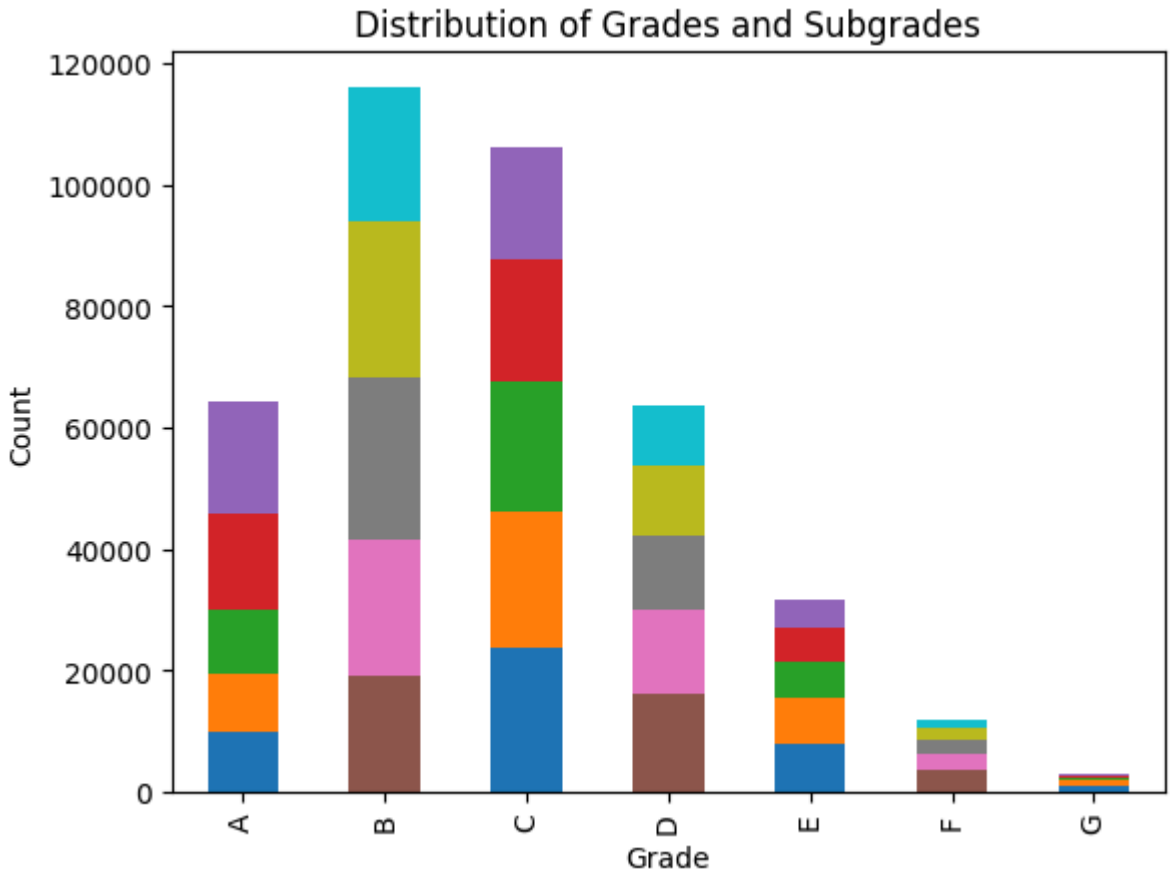
In [ ]:
```
print(f'3/4th of People are paying EMI as less than 500 {np.round(267469 / df.shape[0] * 100,2)}'+' %')
```

```
3/4th of People are paying EMI as less than 500 67.54 %
```

- ***Grades & SubGrades***

In [ ]:
```
grouped = df.groupby(['grade', 'sub_grade']).size().unstack(fill_value=0)
plt.figure(figsize=(10, 6))
grouped.plot(kind='bar', stacked=True,legend=False)
plt.title('Distribution of Grades and Subgrades')
plt.xlabel('Grade')
plt.ylabel('Count')
plt.show()
```

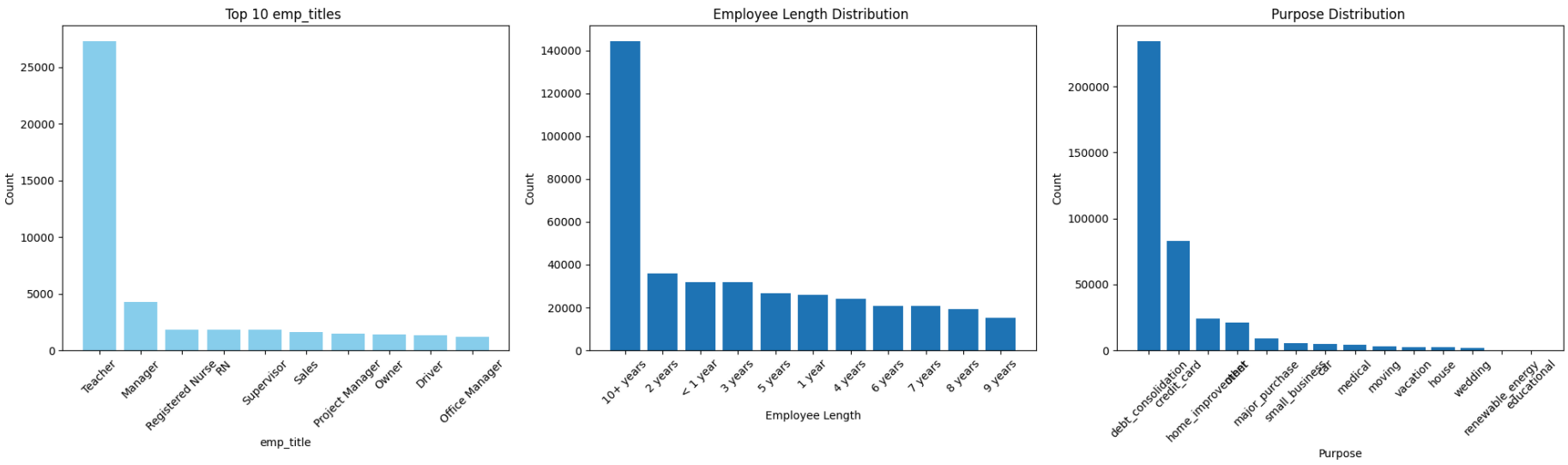
```
<Figure size 1000x600 with 0 Axes>
```



In [ ]:
```
top_emp_titles = df.groupby(['emp_title']).size().sort_values(ascending=False)[:10]
least_emp_titles = df.groupby(['emp_title']).size().sort_values(ascending=True)[:10]
```

In [ ]:

```
fig, axs = plt.subplots(1, 3, figsize=(20, 6))

# Emplotee Titles
axs[0].bar(top_emp_titles.index, top_emp_titles.values, color='skyblue')
axs[0].set_title('Top 10 emp_titles ')
axs[0].set_xlabel('emp_title')
axs[0].set_ylabel('Count')
axs[0].tick_params(axis='x', rotation=45)

# Employee Lengths
axs[1].bar(df['emp_length'].value_counts().index, df['emp_length'].value_counts().values)
axs[1].set_title('Employee Length Distribution ')
axs[1].set_xlabel('Employee Length')
axs[1].set_ylabel('Count')
axs[1].tick_params(axis='x', rotation=45)

#
axs[2].bar(df['purpose'].value_counts().index, df['purpose'].value_counts().values)
axs[2].set_title('Purpose Distribution ')
axs[2].set_xlabel('Purpose')
axs[2].set_ylabel('Count')
axs[2].tick_params(axis='x', rotation=45)


plt.tight_layout()
plt.show()
```



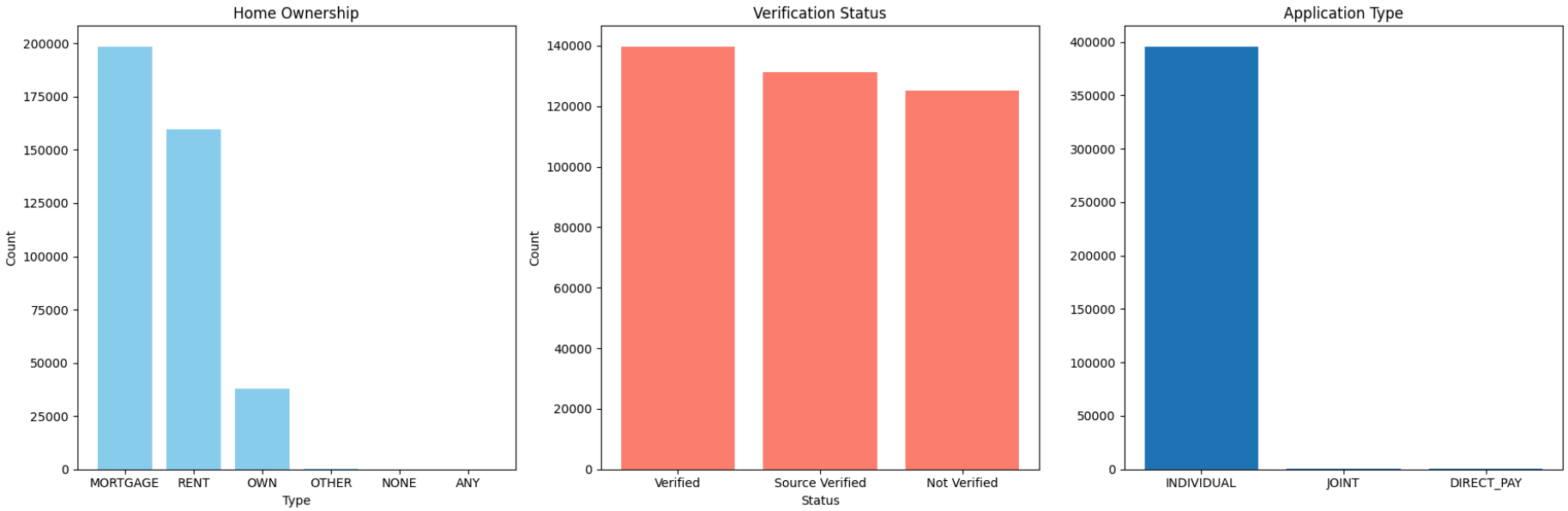
In [ ]:
```
home_ownership_data = df['home_ownership'].value_counts()
application_type_data = df['application_type'].value_counts()
verification_status_data = df['verification_status'].value_counts()
```

In [ ]:
```
home_ownership_data.values
```

Out[ ]:
```
array([198348, 159790,  37746,    112,     31,      3])
```

In [ ]:
```
fig, axs = plt.subplots(1, 3, figsize=(18, 6))

axs[0].bar(home_ownership_data.keys(), home_ownership_data.values, color='skyblue')
axs[1].bar(verification_status_data.keys(), verification_status_data.values, color='salmon')

axs[2].bar(application_type_data.keys(), application_type_data.values)

axs[0].set_title('Home Ownership')
axs[0].set_xlabel('Type')
axs[0].set_ylabel('Count')
axs[1].set_title('Verification Status')
axs[1].set_xlabel('Status')
axs[1].set_ylabel('Count')
axs[2].set_title('Application Type')
```

```
plt.tight_layout()

plt.show()
```



- Converting Mort_acc to booleans

- Note
  - In Loan Business if any person had mort_acc > 4 it is very diffcult to get loan next time so
  - if mort_acc < 4 then 1 else 0

In [ ]:
```
df['mort_acc'] = df['mort_acc'].apply(lambda x: 1 if x < 4 else 0)
```

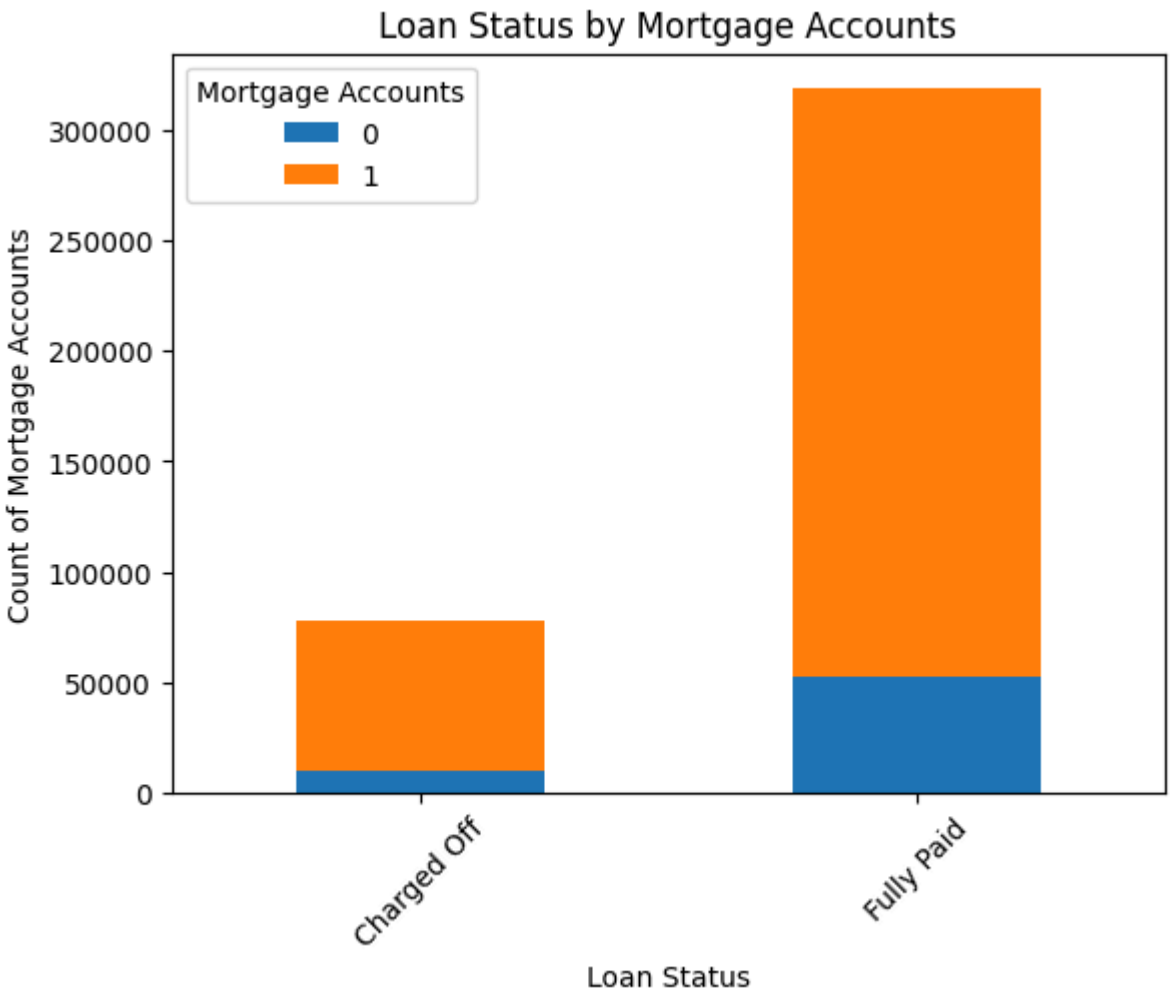
In [ ]:
```
df.head()
```

Out[ ]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	verification_status
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	Not Verified
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	Not Verified
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	Source Verified
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 years	RENT	54000.0	Not Verified
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years	MORTGAGE	55000.0	Verified

In [ ]:

```
grouped_data = df.groupby(['loan_status', 'mort_acc'])['mort_acc'].count().unstack(fill_value=0)
grouped_data.plot(kind='bar', stacked=True)
plt.title('Loan Status by Mortgage Accounts')
plt.xlabel('Loan Status')
plt.ylabel('Count of Mortgage Accounts')
plt.xticks(rotation=45)
plt.legend(title='Mortgage Accounts')
plt.show()
```



In [ ]:
```
danger_accounts = df[df['mort_acc'] == 0].index
danger_accounts
```

Out[ ]:
```
Index([     5,     10,     11,     12,     14,     15,     17,     21,     23,
           29,
       ...
       395962, 395984, 395987, 395999, 396000, 396006, 396015, 396018, 396023,
       396028],
      dtype='int64', length=63202)
```

In [ ]:
```
df[df['loan_status'] == 'Charged Off'].index.isin(danger_accounts).shape[0], df[df['loan_status'] == 'Charged Off'].shape[0]
```

Out[ ]:
```
(77673, 77673)
```

In [ ]:
```
df[df['loan_status'] == 'Charged Off'].shape[0]
```

Out[ ]:
```
77673
```

In [ ]:
```
charged_off_danger_accounts_count = df[df['loan_status'] == 'Charged Off'].index.isin(danger_accounts).sum()
total_charged_off_count = df[df['loan_status'] == 'Charged Off'].shape[0]

print("Count of Charged Off accounts in danger_accounts:", charged_off_danger_accounts_count)
print("Total count of Charged Off accounts:", total_charged_off_count)
```

```
Count of Charged Off accounts in danger_accounts: 10091
Total count of Charged Off accounts: 77673
```

In [ ]:
```
fully_paid_danger_accounts_count = df[df['loan_status'] != 'Charged Off'].index.isin(danger_accounts).sum()
fully_paid_count = df[df['loan_status'] != 'Charged Off'].shape[0]

print("Count of Fully Paid accounts in danger_accounts:", fully_paid_danger_accounts_count)
print("Total count of Fully Paid  accounts:", fully_paid_count)
```

```
Count of Fully Paid accounts in danger_accounts: 53111
Total count of Fully Paid  accounts: 318357
```

In [ ]:
```
print(f'{np.round(charged_off_danger_accounts_count/total_charged_off_count*100,2)} % of Charged Off accounts are in danger_ac
```

```
12.99 % of Charged Off accounts are in danger_accounts
```

In [ ]:
```
print(f'{np.round(fully_paid_danger_accounts_count/fully_paid_count*100,2)} % of Fully Paid accounts are in danger_accounts')
```

```
16.68 % of Fully Paid accounts are in danger_accounts
```

In [ ]:
```
num_df = df[num_col]
num_df.head()
```

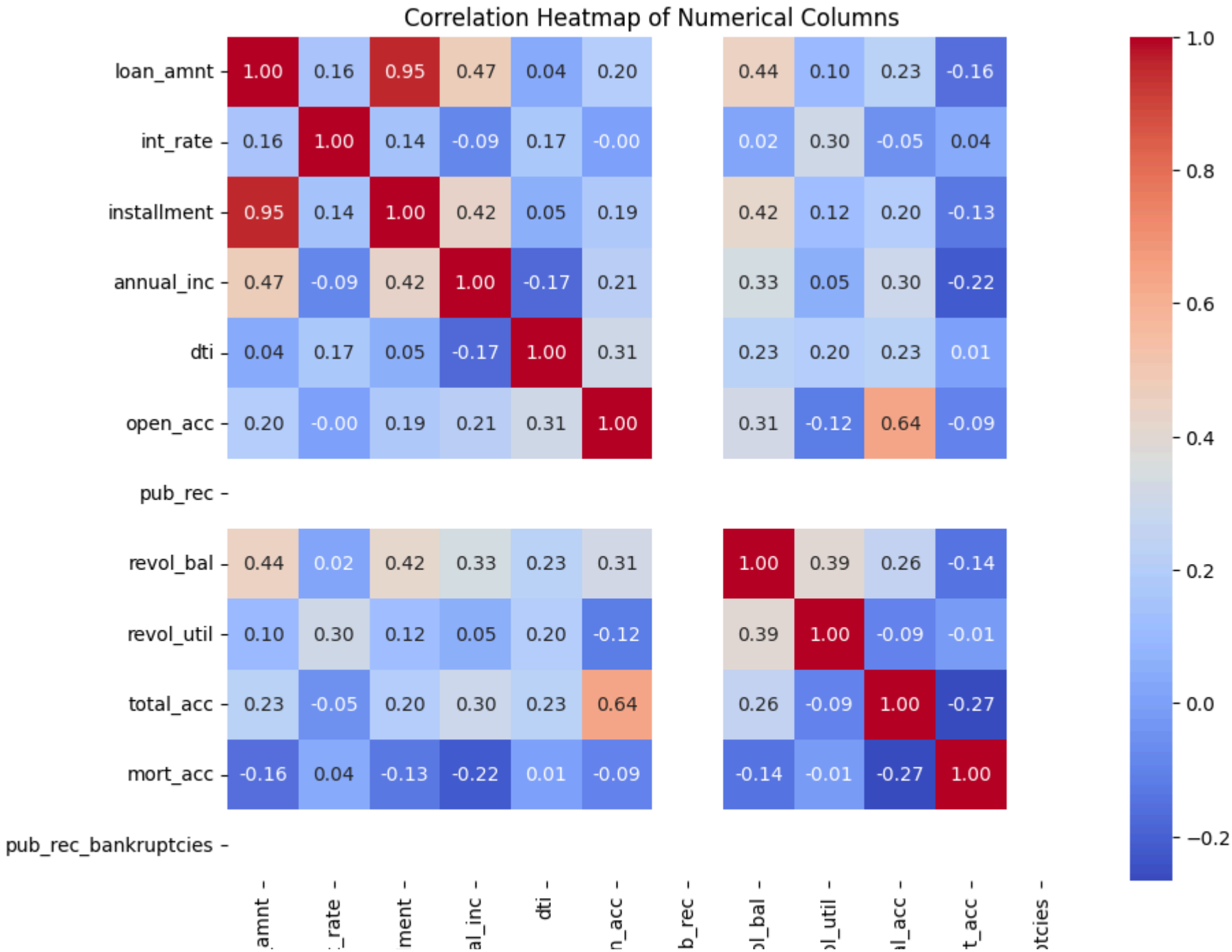
Out[ ]:

	loan_amnt	int_rate	installment	annual_inc	dti	open_acc	pub_rec	revol_bal	revol_util	total_acc	mort_acc	pub_rec_bankruptcies
0	10000.0	11.44	329.48	117000.0	26.24	16.0	0.0	36369.0	41.8	25.0	1	0.0
1	8000.0	11.99	265.68	65000.0	22.05	17.0	0.0	20131.0	53.3	27.0	1	0.0
2	15600.0	10.49	506.97	43057.0	12.79	13.0	0.0	11987.0	92.2	26.0	1	0.0
3	7200.0	6.49	220.65	54000.0	2.60	6.0	0.0	5472.0	21.5	13.0	1	0.0
4	24375.0	17.27	609.33	55000.0	33.95	13.0	0.0	24584.0	69.8	43.0	1	0.0

- Trying to Identify is there any correaltion, pair plot is not helping me

In [ ]:

```
numerical_cols = df[num_col[:5]]
corr_matrix = num_df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of Numerical Columns')
plt.show()
```


Correlation Heatmap of Numerical Columns
loan_amnt
int_rate
installment
annual_inc
dti
open_acc
pub_rec
revol_bal
revol_util
total_acc
mort_acc
pub_rec_bankruptcies
1.00 0.16 0.95 0.47 0.04 0.20 0.44 0.10 0.23 -0.16
0.16 1.00 0.14 -0.09 0.17 -0.00 0.02 0.30 -0.05 0.04
0.95 0.14 1.00 0.42 0.05 0.19 0.42 0.12 0.20 -0.13
0.47 -0.09 0.42 1.00 -0.17 0.21 0.33 0.05 0.30 -0.22
0.04 0.17 0.05 -0.17 1.00 0.31 0.23 0.20 0.23 0.01
0.20 -0.00 0.19 0.21 0.31 1.00 0.31 -0.12 0.64 -0.09
0.44 0.02 0.42 0.33 0.23 0.31 1.00 0.39 0.26 -0.14
0.10 0.30 0.12 0.05 0.20 -0.12 0.39 1.00 -0.09 -0.01
0.23 -0.05 0.20 0.30 0.23 0.64 0.26 -0.09 1.00 -0.27
-0.16 0.04 -0.13 -0.22 0.01 -0.09 -0.14 -0.01 -0.27 1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

In [ ]:
```
num_df.head()
```

Out[ ]:

	loan_amnt	int_rate	installment	annual_inc	dti	open_acc	pub_rec	revol_bal	revol_util	total_acc	mort_acc	pub_rec_bankruptcies
0	10000.0	11.44	329.48	117000.0	26.24	16.0	0.0	36369.0	41.8	25.0	1	0.0
1	8000.0	11.99	265.68	65000.0	22.05	17.0	0.0	20131.0	53.3	27.0	1	0.0
2	15600.0	10.49	506.97	43057.0	12.79	13.0	0.0	11987.0	92.2	26.0	1	0.0
3	7200.0	6.49	220.65	54000.0	2.60	6.0	0.0	5472.0	21.5	13.0	1	0.0
4	24375.0	17.27	609.33	55000.0	33.95	13.0	0.0	24584.0	69.8	43.0	1	0.0

- Notes
  - There is Strong realtionship b/w Installements & Loan Amount
  - There is Strong realtionship b/w pub_rec_bankruptcies & pub_rec
  - There is Strong realtionship b/w open_acc & total_acc

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

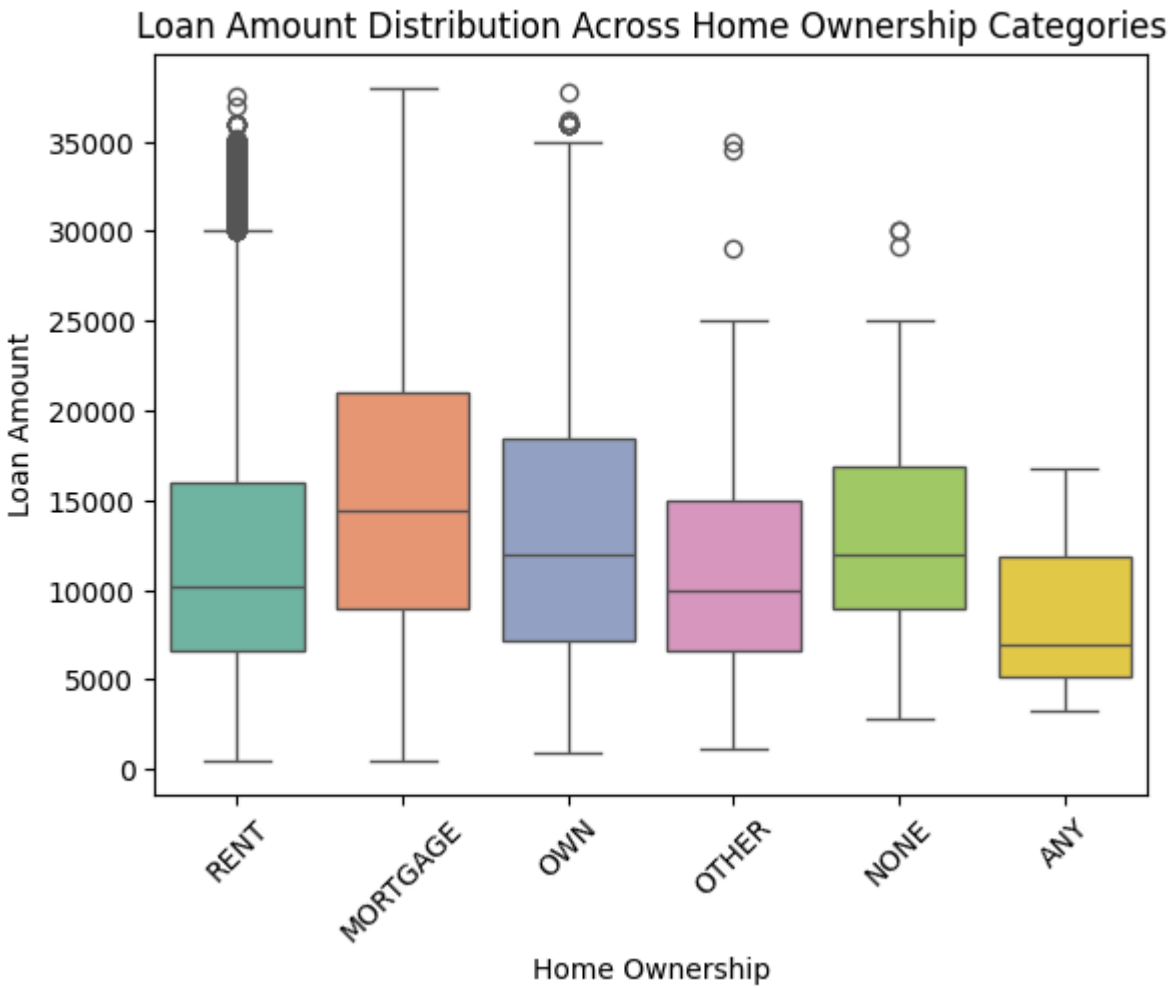
Out[ ]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	verification_status
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	Not Verified
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	Not Verified
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	Source Verified
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 years	RENT	54000.0	Not Verified
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years	MORTGAGE	55000.0	Verified

In [ ]:

```
sns.boxplot(x='home_ownership', y='loan_amnt', data=df, palette='Set2')
plt.title('Loan Amount Distribution Across Home Ownership Categories')
plt.xlabel('Home Ownership')
plt.ylabel('Loan Amount')
plt.xticks(rotation=45)
plt.show()
```

```
<ipython-input-1344-81be35ab8090>:1: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and se
t `legend=False` for the same effect.
```



In [ ]:
```
df_2 = df.copy()
```

- Encoding

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

Out[ ]: (396030, 27)

In [ ]:
```
# performing One hot encoding if we have unique values less than 3
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
```

In [ ]:
```
df['term'].nunique()
```

Out[ ]:
```
2
```

In [ ]:
```
for i in cat_col:
  if df[i].nunique()<=3:
    new_col = i+'_encoded'
    df[new_col] = pd.DataFrame(label_encoder.fit_transform(df[i])).astype('int64')
    df.drop(columns=[i],inplace=True)
```

- grade follows some order so using label encoding

In [ ]:
```
df['grade_encoded'] = pd.DataFrame(label_encoder.fit_transform(df['grade'])).astype('int64')
df.drop(columns=['grade'],inplace=True)
```

In [ ]:
```
df['home_ownership_encoded'] = pd.DataFrame(label_encoder.fit_transform(df['home_ownership'])).astype('int64')
df.drop(columns=['home_ownership'],inplace=True)
```

In [ ]:
```
df['emp_length_encoded'] = pd.DataFrame(label_encoder.fit_transform(df['emp_length'])).astype('int64')
df.drop(columns=['emp_length'],inplace=True)
```

In [ ]:
```
df['purpose_encoded'] = pd.DataFrame(label_encoder.fit_transform(df['purpose'])).astype('int64')
df.drop(columns=['purpose'],inplace=True)
```

In [ ]:
```
df['sub_grade_encoded'] = pd.DataFrame(label_encoder.fit_transform(df['sub_grade'])).astype('int64')
df.drop(columns=['sub_grade'],inplace=True)
```

In [ ]:
```
cat_col = df.select_dtypes(include=['object']).columns
cat_col
```

Out[ ]:
```
Index(['emp_title', 'issue_d', 'title', 'earliest_cr_line', 'state'], dtype='object')
```

In [ ]:
```
# Target Encoding for above columns
```

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

Out[ ]:
```
(396030, 27)
```

In [ ]:
```
lc_mean = df.groupby('emp_title')['loan_status_encoded'].mean()
df['emp_title_encoded'] = df['emp_title'].map(lc_mean)
```

In [ ]:
```
df.drop(columns=['emp_title'],inplace=True)
```

In [ ]:
```
title_mean = df.groupby('title')['loan_status_encoded'].mean()
df['title_encoded'] = df['title'].map(title_mean)
```

In [ ]:
```
df.drop(columns=['title'],inplace=True)
```

In [ ]:
```
state_mean = df.groupby('state')['loan_status_encoded'].mean()
df['state_encoded'] = df['state'].map(state_mean)
```

In [ ]:
```
df.drop(columns=['state'],inplace=True)
```

In [ ]:
```
# Date values Encoding
```

In [ ]:
```
df['issue_date'] = pd.to_datetime(df['issue_d'])
# Extract month and year components
df['issue_month'] = df['issue_date'].dt.month
df['issue_year'] = df['issue_date'].dt.year
df.drop(columns=['issue_d','issue_date'],inplace=True)
```

```
<ipython-input-1365-06eb8a3d5c4b>:1: UserWarning:

Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consisten
t and as-expected, please specify a format.
```

In [ ]:
```
df['ecr_date'] = pd.to_datetime(df['earliest_cr_line'])
# Extract month and year components
```

```
df['ecr_date_month'] = df['ecr_date'].dt.month
df['ecr_date_year'] = df['ecr_date'].dt.year
df.drop(columns=['ecr_date','earliest_cr_line'],inplace=True)
```

```
<ipython-input-1366-1eedf690f98d>:1: UserWarning:

Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consisten
t and as-expected, please specify a format.
```

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

Out[ ]:
```
(396030, 29)
```

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

Out[ ]:

	loan_amnt	int_rate	installment	annual_inc	dti	open_acc	pub_rec	revol_bal	revol_util	total_acc	mort_acc	pub_rec_bankruptcies
0	10000.0	11.44	329.48	117000.0	26.24	16.0	0.0	36369.0	41.8	25.0	1	0.0
1	8000.0	11.99	265.68	65000.0	22.05	17.0	0.0	20131.0	53.3	27.0	1	0.0
2	15600.0	10.49	506.97	43057.0	12.79	13.0	0.0	11987.0	92.2	26.0	1	0.0
3	7200.0	6.49	220.65	54000.0	2.60	6.0	0.0	5472.0	21.5	13.0	1	0.0
4	24375.0	17.27	609.33	55000.0	33.95	13.0	0.0	24584.0	69.8	43.0	1	0.0

5 rows × 29 columns

In [ ]:
```
df['issue_month'].nunique(),df['issue_year'].nunique(),df['ecr_date_month'].nunique(),df['ecr_date_year'].nunique()
```

Out[ ]:
```
(12, 10, 12, 65)
```

In [ ]:
```
# for ecr_date_year Target_encoding rest all are already label encoded
yr_mean = df.groupby('ecr_date_year')['loan_status_encoded'].mean()
df['ecr_date_year_encoded'] = df['ecr_date_year'].map(yr_mean)
df.drop(columns=['ecr_date_year'],inplace=True)
```

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

Out[ ]:

	loan_amnt	int_rate	installment	annual_inc	dti	open_acc	pub_rec	revol_bal	revol_util	total_acc	mort_acc	pub_rec_bankruptcies
0	10000.0	11.44	329.48	117000.0	26.24	16.0	0.0	36369.0	41.8	25.0	1	0.0
1	8000.0	11.99	265.68	65000.0	22.05	17.0	0.0	20131.0	53.3	27.0	1	0.0
2	15600.0	10.49	506.97	43057.0	12.79	13.0	0.0	11987.0	92.2	26.0	1	0.0
3	7200.0	6.49	220.65	54000.0	2.60	6.0	0.0	5472.0	21.5	13.0	1	0.0
4	24375.0	17.27	609.33	55000.0	33.95	13.0	0.0	24584.0	69.8	43.0	1	0.0

5 rows × 29 columns

In [ ]:
```
df.isnull().sum()
```

Out[ ]:

```
loan_amnt                       191
int_rate                       3777
installment                   11250
annual_inc                    16700
dti                             275
open_acc                      10307
pub_rec                       57758
revol_bal                     21259
revol_util                      288
total_acc                      8499
mort_acc                          0
pub_rec_bankruptcies          45650
term_encoded                      0
verification_status_encoded       0
loan_status_encoded               0
initial_list_status_encoded       0
application_type_encoded          0
grade_encoded                     0
home_ownership_encoded            0
emp_length_encoded                0
purpose_encoded                   0
sub_grade_encoded                 0
emp_title_encoded                 0
title_encoded                  1756
state_encoded                     0
issue_month                       0
issue_year                        0
ecr_date_month                    0
ecr_date_year_encoded             0
dtype: int64
```

In [ ]:

```
for i in df.columns:
  if df[i].isnull().sum()>0:
    df[i].fillna(df[i].median(),inplace=True)
```

In [ ]:

```
df.isnull().sum()
```

Out[ ]:
```
loan_amnt                      0
int_rate                       0
installment                    0
annual_inc                     0
dti                            0
open_acc                       0
pub_rec                        0
revol_bal                      0
revol_util                     0
total_acc                      0
mort_acc                       0
pub_rec_bankruptcies           0
term_encoded                   0
verification_status_encoded    0
loan_status_encoded            0
initial_list_status_encoded    0
application_type_encoded       0
grade_encoded                  0
home_ownership_encoded         0
emp_length_encoded             0
purpose_encoded                0
sub_grade_encoded              0
emp_title_encoded              0
title_encoded                  0
state_encoded                  0
issue_month                    0
issue_year                     0
ecr_date_month                 0
ecr_date_year_encoded          0
dtype: int64
```

3. Logistic Regression Model

In [ ]:
```
# Required Libraries
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, roc_auc_score, roc_curv
```

In [ ]:
```
x = df.drop(columns=['loan_status_encoded'])
y = df['loan_status_encoded']
```

In [ ]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

In [ ]:
```
standardScaler = StandardScaler()
x_train = standardScaler.fit_transform(x_train)
x_test = standardScaler.transform(x_test)
```

In [ ]:
```
x_train.shape, x_test.shape
```

Out[ ]:
```
((316824, 28), (79206, 28))
```

In [ ]:
```
model = LogisticRegression()
model.fit(x_train, y_train)
```

Out[ ]:
```
▾ LogisticRegression
LogisticRegression()
```

In [ ]:
```
y_pred = model.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)
```

In [ ]:
```
print(f"Accuracy : {np.round(100*accuracy,2)}%")
```

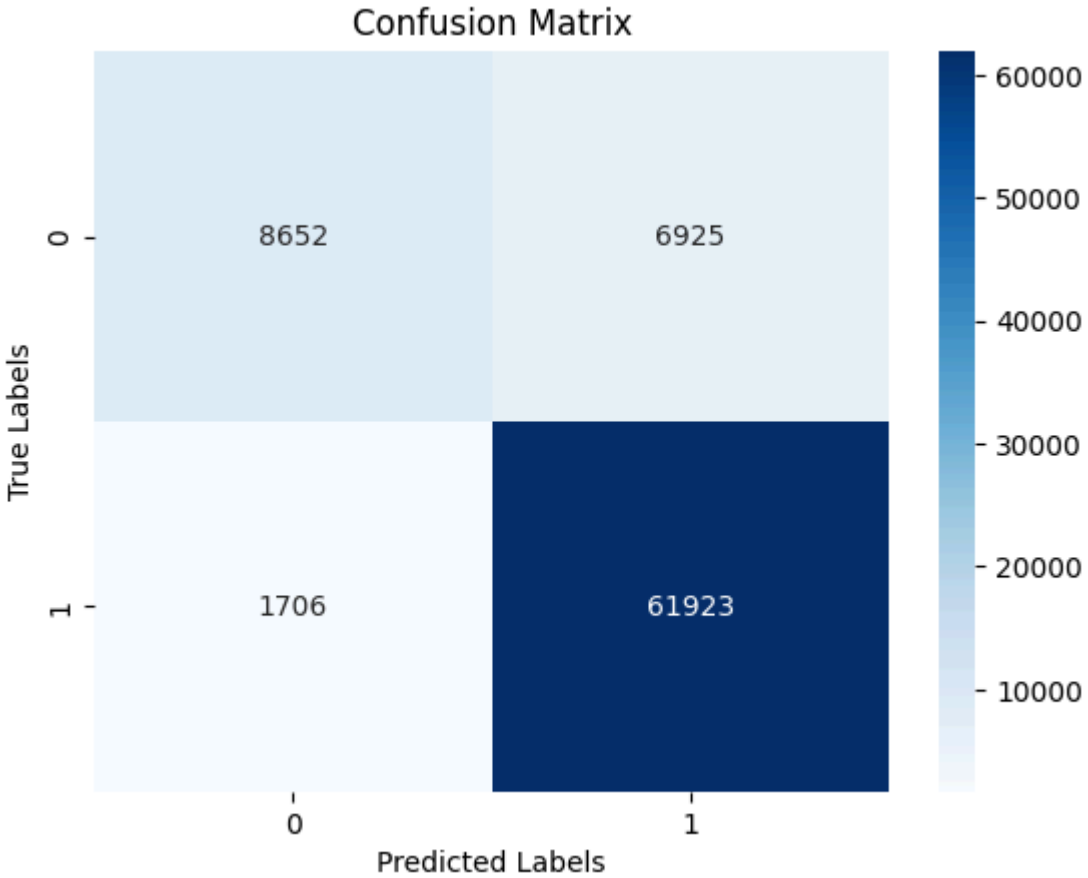
```
Accuracy : 89.1%
```

In [ ]:
```
print(f"precision : {np.round(100*precision,2)}%")
```

```
precision : 89.94%
```

In [ ]:
```
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted Labels')
```

```
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```



In [ ]:
```
print(f"recall : {np.round(100*recall,2)}")
```

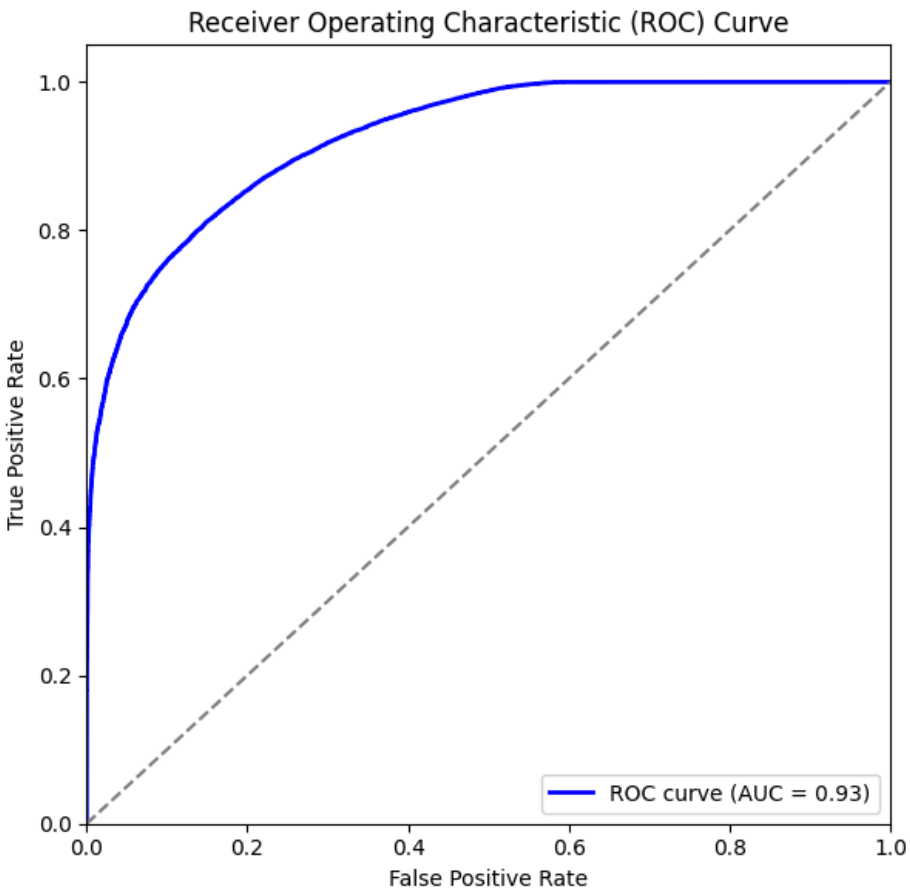
```
recall : 97.32
```

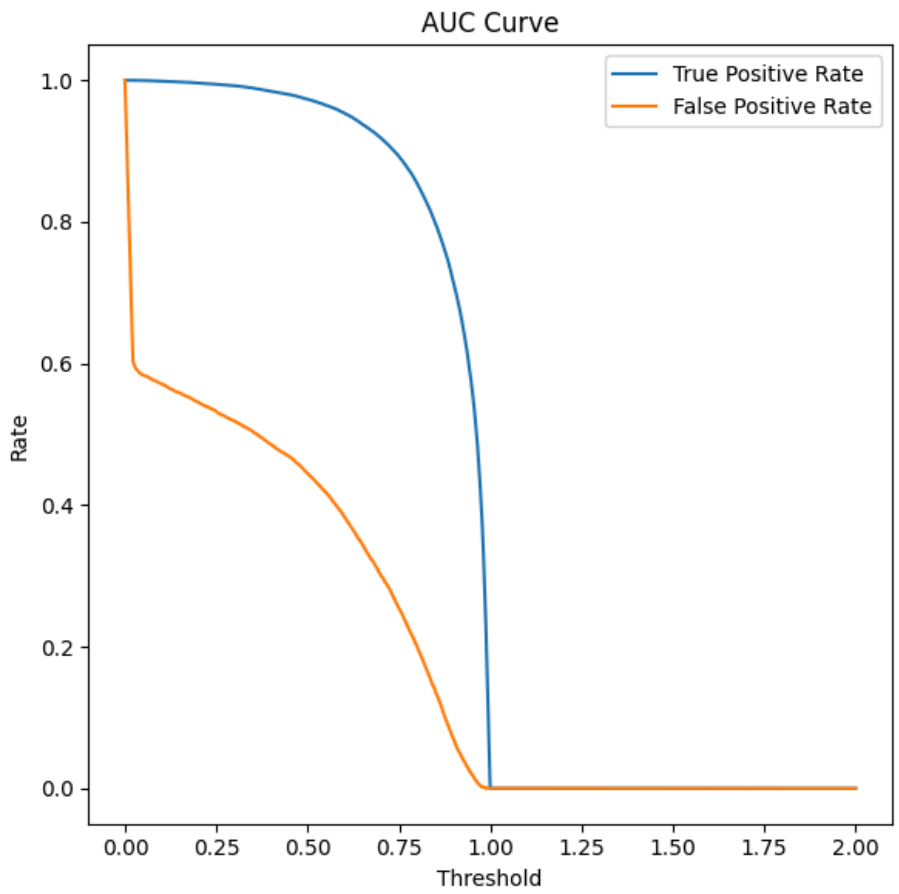
In [ ]:
```
print(f"f1_score : {np.round(100*f1,2)}")
```

```
f1_score : 93.48
```

In [ ]:
```
from sklearn.metrics import roc_curve, auc
y_pred_proba = model.predict_proba(x_test)[:, 1]  # Probability of positive class
```

```python
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
roc_auc = auc(fpr, tpr)


# ROC
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (AUC = {:.2f})'.format(roc_auc))
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')

# Plot AUC curve
plt.subplot(1, 2, 2)
plt.plot(thresholds, tpr, label='True Positive Rate')
plt.plot(thresholds, fpr, label='False Positive Rate')
plt.xlabel('Threshold')
plt.ylabel('Rate')
plt.title('AUC Curve')
plt.legend()


plt.tight_layout()
plt.show()
```





5. Actionable Insights & Recommendations

5.1 The percentage of customers who have fully paid their loan

In [ ]:
```
fully_paid_percentage = (df_2['loan_status'] == 'Fully Paid').mean() * 100
print(f"The percentage of customers who have fully paid their loan: {fully_paid_percentage:.2f}%")
```

```
The percentage of customers who have fully paid their loan: 80.39%
```

5.2 Correlation Between Loan Amount & Installment

In [ ]:
```
correlation = df_2['loan_amnt'].corr(df_2['installment'])
print(f"The correlation between Loan Amount and Installment: {100 * correlation:.2f} %")
```

```
The correlation between Loan Amount and Installment: 95.06 %
```

5.3 Majority_home_ownership

In [ ]:
```
majority_home_ownership = df_2['home_ownership'].value_counts().idxmax()
MORTGAGE_percentage = np.round(198348 / df_2.shape[0] *100,2) # Here 198348 is sum of mortgae count
print(f"The majority of people have home ownership as: {majority_home_ownership} with {MORTGAGE_percentage} %")
```

```
The majority of people have home ownership as: MORTGAGE with 50.08 %
```

5.4 A Grade People Likely to Full Pay their Loan

In [ ]:
```
fully_paid_grade_A_percentage = (df_2[df_2['grade'] == 'A']['loan_status'] == 'Fully Paid').mean()
overall_fully_paid_percentage = (df_2['loan_status'] == 'Fully Paid').mean()

is_grade_A_more_likely = fully_paid_grade_A_percentage > overall_fully_paid_percentage
print(f"People with grades 'A' are more likely to fully pay their loan: {is_grade_A_more_likely}")
```

```
People with grades 'A' are more likely to fully pay their loan: True
```

5.5 Top 2 Job Titles

In [ ]:
```
top_2_job_titles = df_2['emp_title'].value_counts().head(2).index.tolist()
print(f"The top 2 afforded job titles are: {top_2_job_titles}")
```

```
The top 2 afforded job titles are: ['Teacher', 'Manager']
```

In [ ]: