Case Study (Loan Tap)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data= pd.read_csv("loan_tap.csv")
data.head()
```

```
   loan_amnt        term  int_rate  installment grade sub_grade  \
0    10000.0   36 months     11.44       329.48     B        B4
1     8000.0   36 months     11.99       265.68     B        B5
2    15600.0   36 months     10.49       506.97     B        B3
3     7200.0   36 months      6.49       220.65     A        A2
4    24375.0   60 months     17.27       609.33     C        C5

                 emp_title emp_length home_ownership  annual_inc  ...  \
0                Marketing  10+ years           RENT    117000.0  ...
1          Credit analyst     4 years       MORTGAGE     65000.0  ...
2             Statistician   < 1 year           RENT     43057.0  ...
3           Client Advocate    6 years           RENT     54000.0  ...
4  Destiny Management Inc.    9 years       MORTGAGE     55000.0  ...

  open_acc pub_rec revol_bal revol_util total_acc  initial_list_status  \
0     16.0     0.0   36369.0       41.8      25.0                    w
1     17.0     0.0   20131.0       53.3      27.0                    f
2     13.0     0.0   11987.0       92.2      26.0                    f
3      6.0     0.0    5472.0       21.5      13.0                    f
4     13.0     0.0   24584.0       69.8      43.0                    f

  application_type  mort_acc  pub_rec_bankruptcies  \
0       INDIVIDUAL       0.0                   0.0
```

```
1       INDIVIDUAL       3.0                   0.0
2       INDIVIDUAL       0.0                   0.0
3       INDIVIDUAL       0.0                   0.0
4       INDIVIDUAL       1.0                   0.0

                                              address
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

[5 rows x 27 columns]
```

---

1. Define problem statement and perform Exploratory Data Analysis

```
Given a set of attributes for an Individual, determine if a credit line should be extended to them. If
so, what should the repayment terms be in business recommendations?
```

a. Observations on shape of data and data types of all attributes

```
target_variable=data['loan_status']

data.shape

(396030, 27)

data.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
```

b. Check for missing value (if any)

```
data.isnull().sum()
```

```
loan_amnt                   0
term                        0
int_rate                    0
installment                 0
grade                       0
sub_grade                   0
emp_title               22927
emp_length              18301
home_ownership              0
annual_inc                  0
verification_status         0
issue_d                     0
loan_status                 0
purpose                     0
title                    1756
dti                         0
earliest_cr_line            0
open_acc                    0
pub_rec                     0
revol_bal                   0
revol_util                276
total_acc                   0
initial_list_status         0
application_type            0
mort_acc                37795
pub_rec_bankruptcies      535
address                     0
dtype: int64
```

c. Display the statistical summary

```
data.describe()
```

	loan_amnt	int_rate	installment	annual_inc
count	396030.000000	396030.000000	396030.000000	3.960300e+05
mean	14113.888089	13.639400	431.849698	7.420318e+04
std	8357.441341	4.472157	250.727790	6.163762e+04
min	500.000000	5.320000	16.080000	0.000000e+00
25%	8000.000000	10.490000	250.330000	4.500000e+04
50%	12000.000000	13.330000	375.430000	6.400000e+04
75%	20000.000000	16.490000	567.300000	9.000000e+04
max	40000.000000	30.990000	1533.810000	8.706582e+06

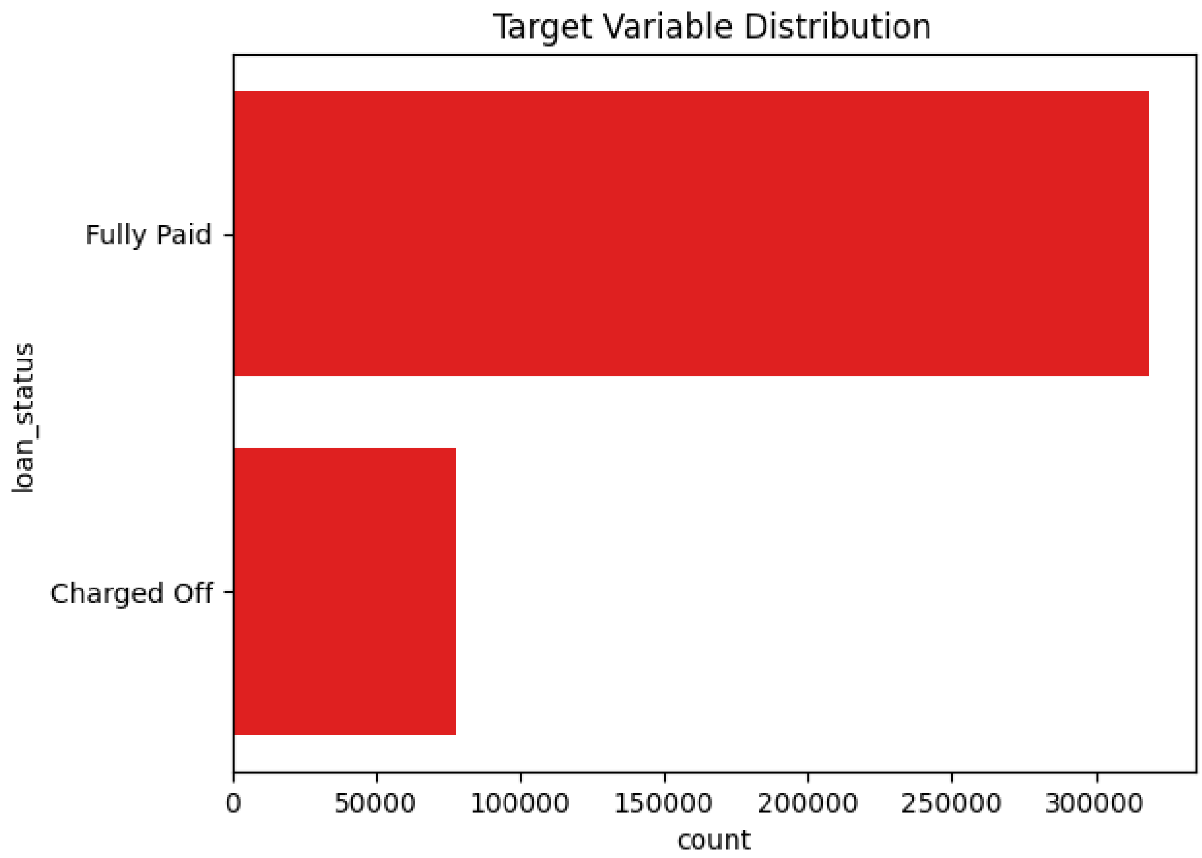
	dti	open_acc	pub_rec	revol_bal
count	396030.000000	396030.000000	396030.000000	3.960300e+05
mean	17.379514	11.311153	0.178191	1.584454e+04
std	18.019092	5.137649	0.530671	2.059184e+04
min	0.000000	0.000000	0.000000	0.000000e+00
25%	11.280000	8.000000	0.000000	6.025000e+03
50%	16.910000	10.000000	0.000000	1.118100e+04
75%	22.980000	14.000000	0.000000	1.962000e+04
max	9999.000000	90.000000	86.000000	1.743266e+06

	revol_util	total_acc	mort_acc	pub_rec_bankruptcies
count	395754.000000	396030.000000	358235.000000	395495.000000
mean	53.791749	25.414744	1.813991	0.121648
std	24.452193	11.886991	2.147930	0.356174
min	0.000000	2.000000	0.000000	0.000000
25%	35.800000	17.000000	0.000000	0.000000
50%	54.800000	24.000000	1.000000	0.000000
75%	72.900000	32.000000	3.000000	0.000000
max	892.300000	151.000000	34.000000	8.000000

d. Univariate Analysis and Bivariate Analysis of all the attributes

```
print(f"\nTarget Variable Distribution:\n{target_variable.value_counts()}")
sns.countplot(target_variable, color='red')
plt.title("Target Variable Distribution")
plt.show()
```

```
Target Variable Distribution:
loan_status
Fully Paid     318357
Charged Off     77673
Name: count, dtype: int64
```

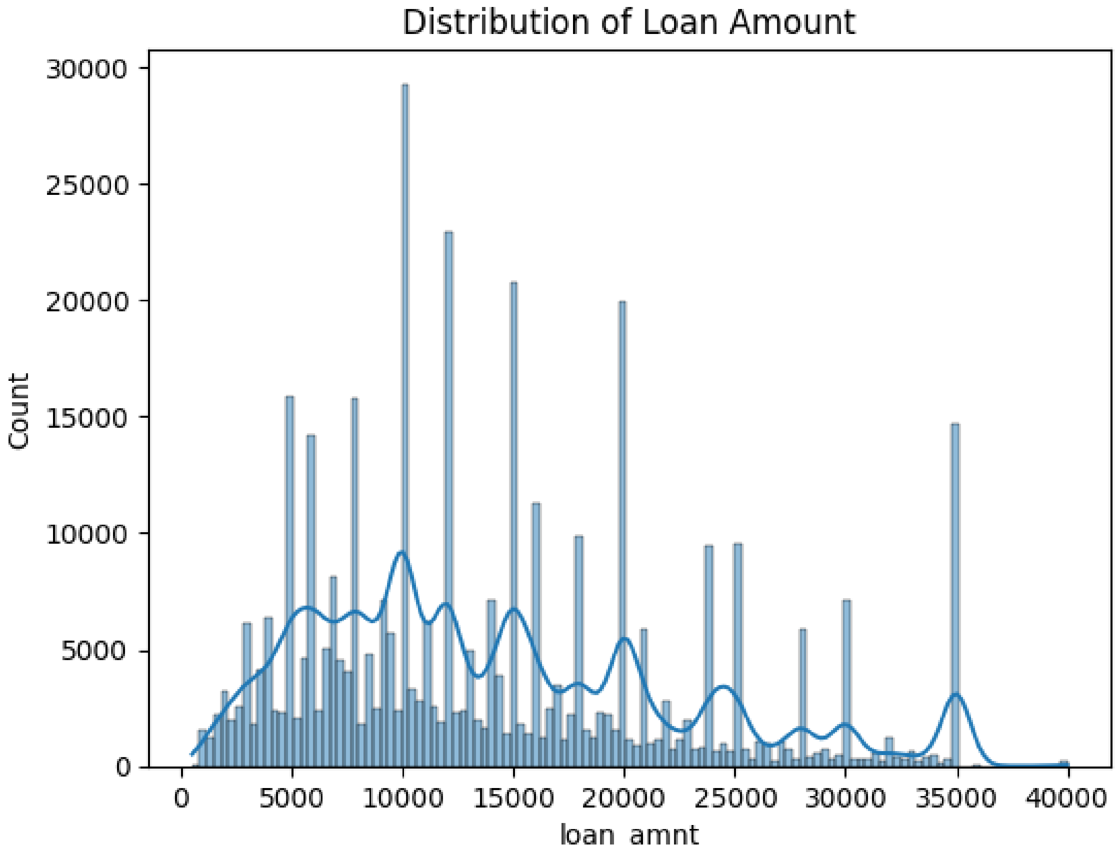


Univeriate

```
numerical_data= data.select_dtypes(include=['int64','float64']).columns
categorical_data=data.select_dtypes(include='object').columns

data[numerical_data]

        loan_amnt  int_rate  installment  annual_inc    dti  open_acc  \
0         10000.0     11.44       329.48    117000.0  26.24      16.0
```

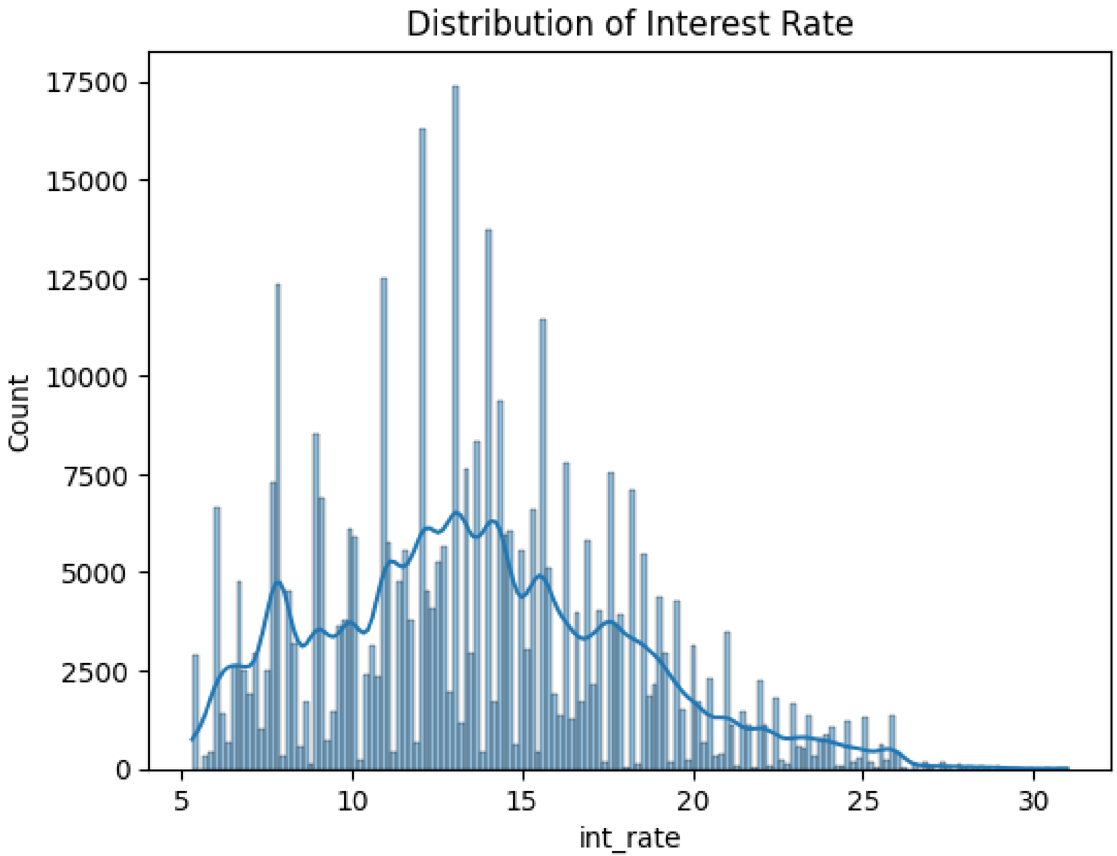
```
1          8000.0     11.99       265.68     65000.0  22.05      17.0
2         15600.0     10.49       506.97     43057.0  12.79      13.0
3          7200.0      6.49       220.65     54000.0   2.60       6.0
4         24375.0     17.27       609.33     55000.0  33.95      13.0
...           ...       ...          ...         ...    ...       ...
396025    10000.0     10.99       217.38     40000.0  15.63       6.0
396026    21000.0     12.29       700.42    110000.0  21.45       6.0
396027     5000.0      9.99       161.32     56500.0  17.56      15.0
396028    21000.0     15.31       503.02     64000.0  15.88       9.0
396029     2000.0     13.61        67.98     42996.0   8.32       3.0

        pub_rec  revol_bal  revol_util  total_acc  mort_acc  \
0           0.0    36369.0        41.8       25.0       0.0
1           0.0    20131.0        53.3       27.0       3.0
2           0.0    11987.0        92.2       26.0       0.0
3           0.0     5472.0        21.5       13.0       0.0
4           0.0    24584.0        69.8       43.0       1.0
...         ...        ...         ...        ...       ...
396025      0.0     1990.0        34.3       23.0       0.0
396026      0.0    43263.0        95.7        8.0       1.0
396027      0.0    32704.0        66.9       23.0       0.0
396028      0.0    15704.0        53.8       20.0       5.0
396029      0.0     4292.0        91.3       19.0       NaN

        pub_rec_bankruptcies
0                        0.0
1                        0.0
2                        0.0
3                        0.0
4                        0.0
...                      ...
396025                   0.0
396026                   0.0
396027                   0.0
```

```
396028                   0.0
396029                   0.0

[396030 rows x 12 columns]

sns.histplot(data['loan_amnt'], kde=True)
plt.title('Distribution of Loan Amount')
plt.show()
```

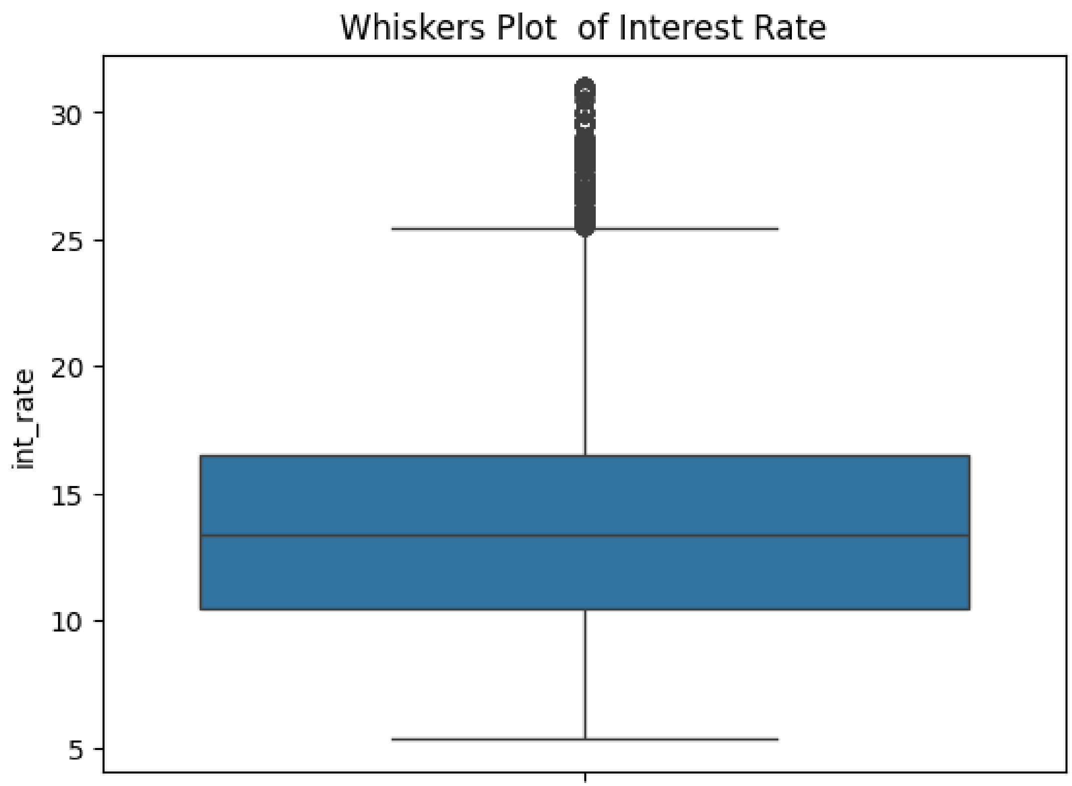


The Loan amount is mostly segregated in the range of 5000-20000 No need to check for outliers in this feature

```
sns.histplot(data['int_rate'], kde=True)
plt.title('Distribution of Interest Rate')
plt.show()
```

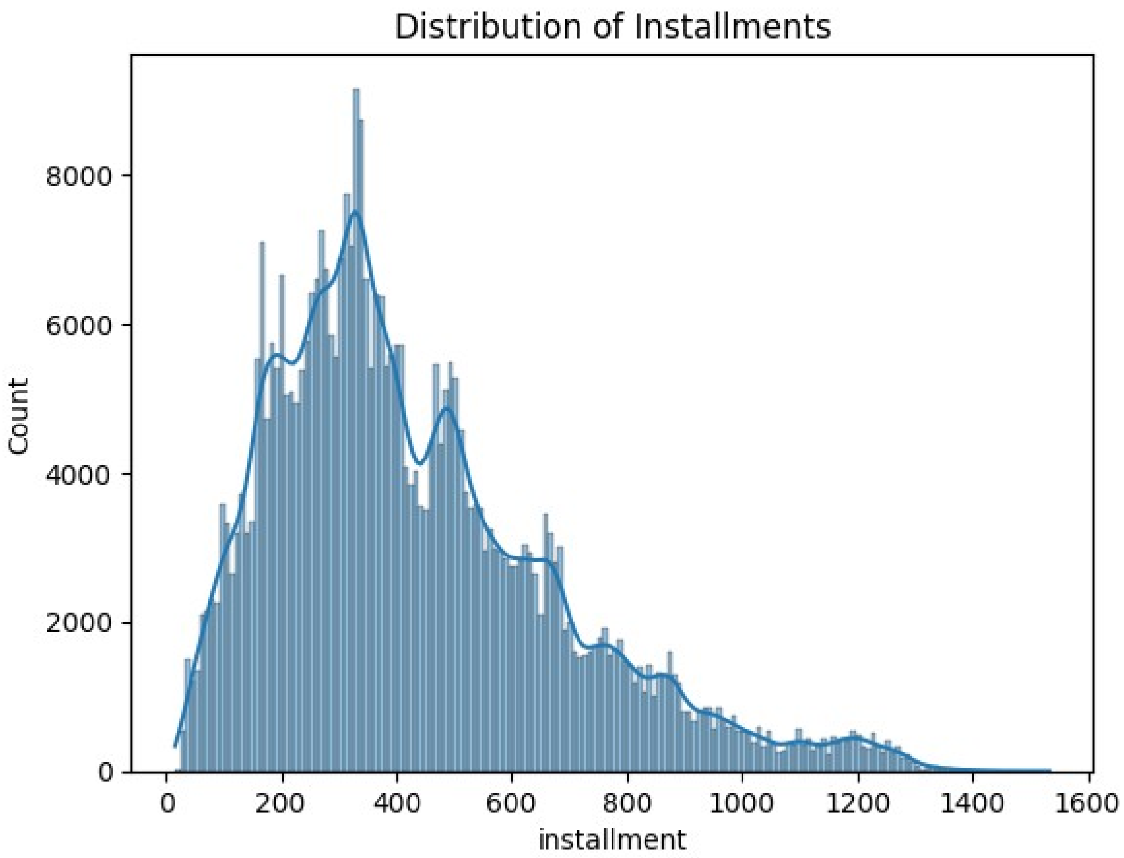


```
sns.boxplot(data=data['int_rate'])
plt.title('Whiskers Plot  of Interest Rate')
plt.show()
```



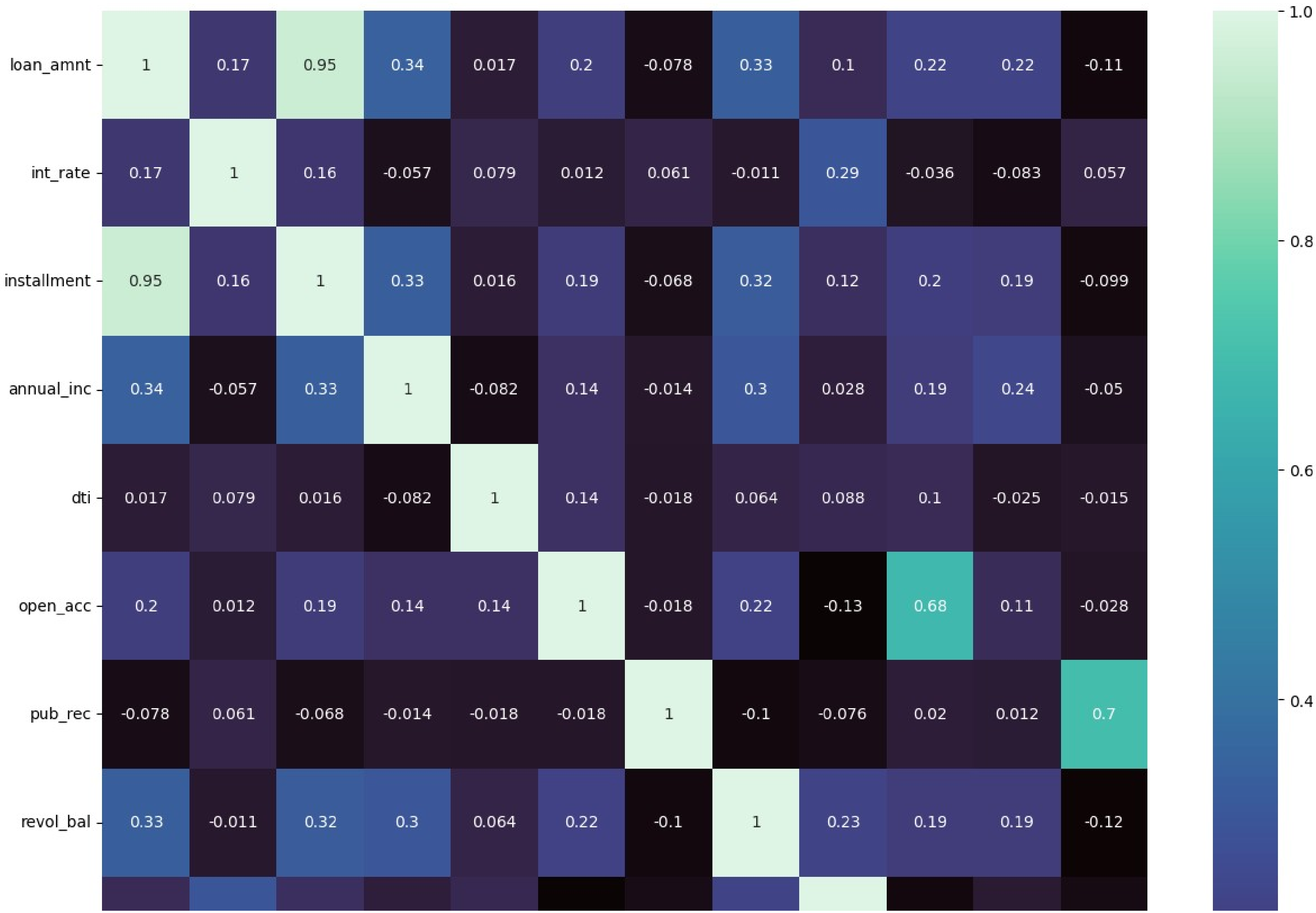
Their are numerous outliers in the interest rate

```
sns.histplot(data['installment'], kde=True)
plt.title('Distribution of Installments')
plt.show()
```



Correlation Analysis (Numerical Features)

```
plt.figure(figsize=(15,15))
sns.heatmap(data=data[numerical_data].corr(), annot=True, cmap=sns.color_palette("mako", as_cmap=True))
plt.show()
```



loan_amnt
int_rate
installment
annual_inc
dti
open_acc
pub_rec
revol_bal
1 0.17 0.95 0.34 0.017 0.2 -0.078 0.33 0.1 0.22 0.22 -0.11
0.17 1 0.16 -0.057 0.079 0.012 0.061 -0.011 0.29 -0.036 -0.083 0.057
0.95 0.16 1 0.33 0.016 0.19 -0.068 0.32 0.12 0.2 0.19 -0.099
0.34 -0.057 0.33 1 -0.082 0.14 -0.014 0.3 0.028 0.19 0.24 -0.05
0.017 0.079 0.016 -0.082 1 0.14 -0.018 0.064 0.088 0.1 -0.025 -0.015
0.2 0.012 0.19 0.14 0.14 1 -0.018 0.22 -0.13 0.68 0.11 -0.028
-0.078 0.061 -0.068 -0.014 -0.018 -0.018 1 -0.1 -0.076 0.02 0.012 0.7
0.33 -0.011 0.32 0.3 0.064 0.22 -0.1 1 0.23 0.19 0.19 -0.12
1.0
0.8
0.6
0.4

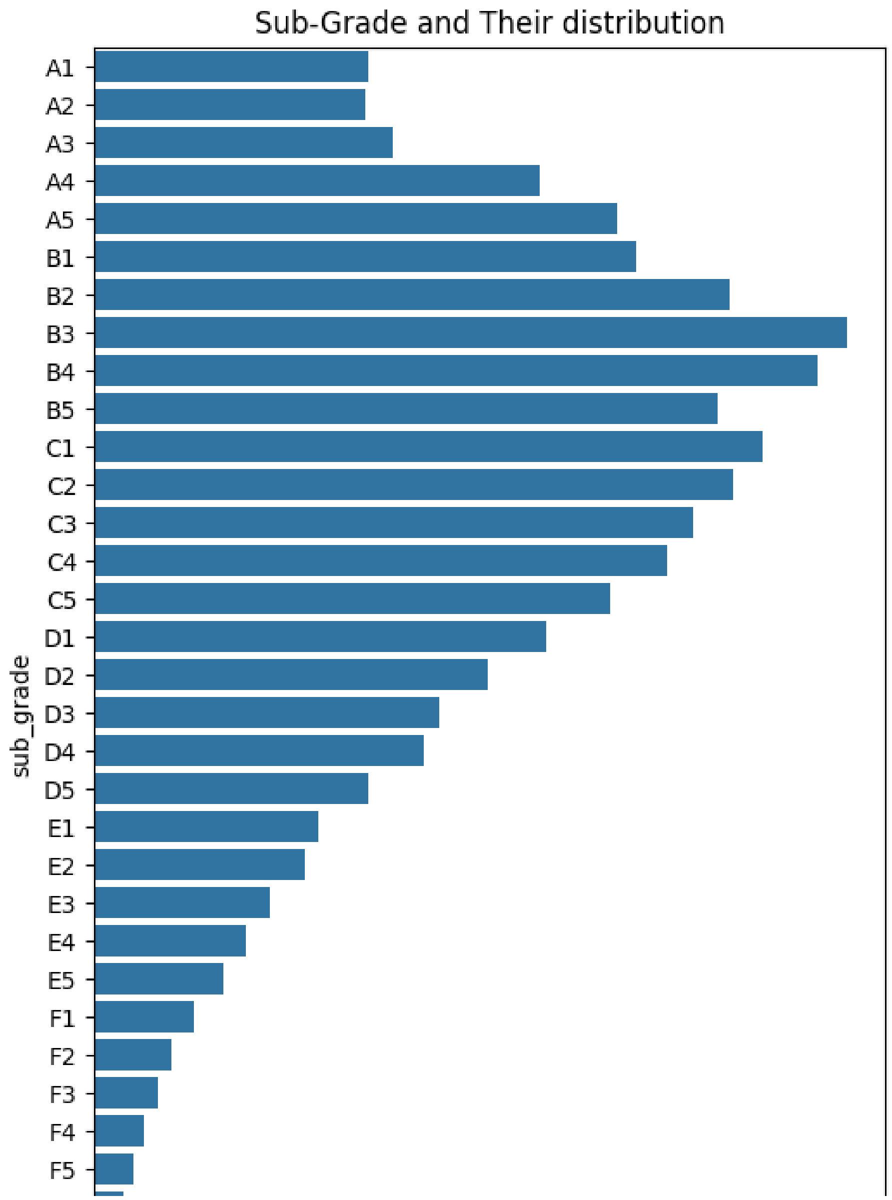
Bivariate

```
data[categorical_data].head(6)
```

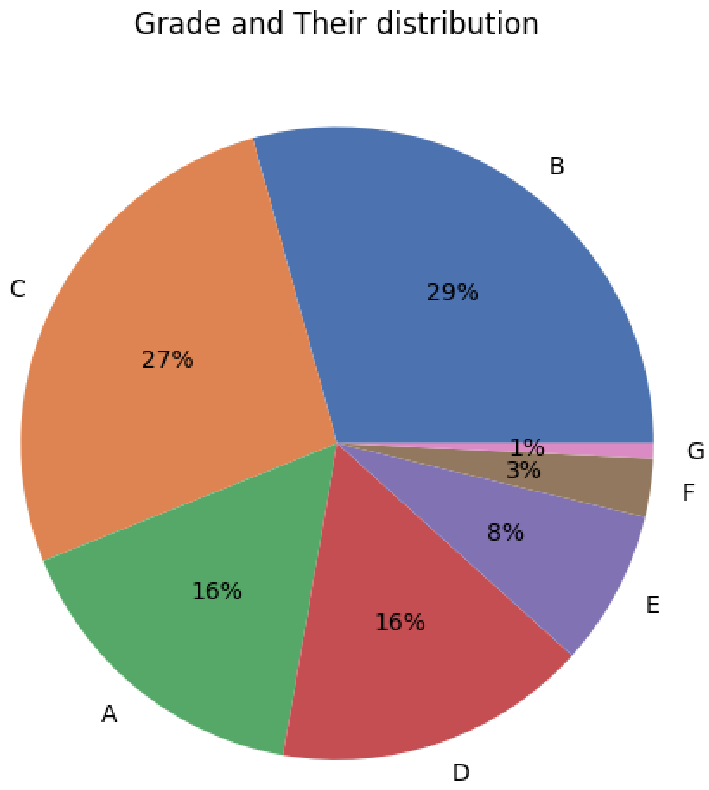
```
        term grade sub_grade                 emp_title emp_length  \
0  36 months     B        B4                 Marketing  10+ years
1  36 months     B        B5           Credit analyst     4 years
2  36 months     B        B3               Statistician   < 1 year
3  36 months     A        A2           Client Advocate    6 years
4  60 months     C        C5  Destiny Management Inc.     9 years
5  36 months     C        C3             HR Specialist  10+ years

  home_ownership verification_status   issue_d  loan_status  \
0           RENT        Not Verified  Jan-2015   Fully Paid
1       MORTGAGE        Not Verified  Jan-2015   Fully Paid
2           RENT     Source Verified  Jan-2015   Fully Paid
3           RENT        Not Verified  Nov-2014   Fully Paid
4       MORTGAGE            Verified  Apr-2013  Charged Off
5       MORTGAGE            Verified  Sep-2015   Fully Paid

             purpose                    title earliest_cr_line  \
0           vacation                 Vacation         Jun-1990
1  debt_consolidation       Debt consolidation         Jul-2004
2         credit_card  Credit card refinancing         Aug-2007
3         credit_card  Credit card refinancing         Sep-2006
4         credit_card    Credit Card Refinance         Mar-1999
5  debt_consolidation       Debt consolidation         Jan-2005

  initial_list_status application_type  \
0                   w       INDIVIDUAL
1                   f       INDIVIDUAL
2                   f       INDIVIDUAL
3                   f       INDIVIDUAL
```

```
4                   f       INDIVIDUAL
5                   f       INDIVIDUAL

                                                address
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
5  1726 Cooper Passage Suite 129\r\nNorth Deniseb...
```

```
plt.figure(figsize=(13,10))

plt.subplot(1,2,2)
plt.pie(data['grade'].value_counts().values, labels=data['grade'].value_counts().index,
colors=sns.color_palette('deep'), autopct='%.0f%%')
plt.title('Grade and Their distribution')

plt.subplot(1,2,1)
sns.countplot(data['sub_grade'].sort_values())
plt.title('Sub-Grade and Their distribution')

plt.show()
```

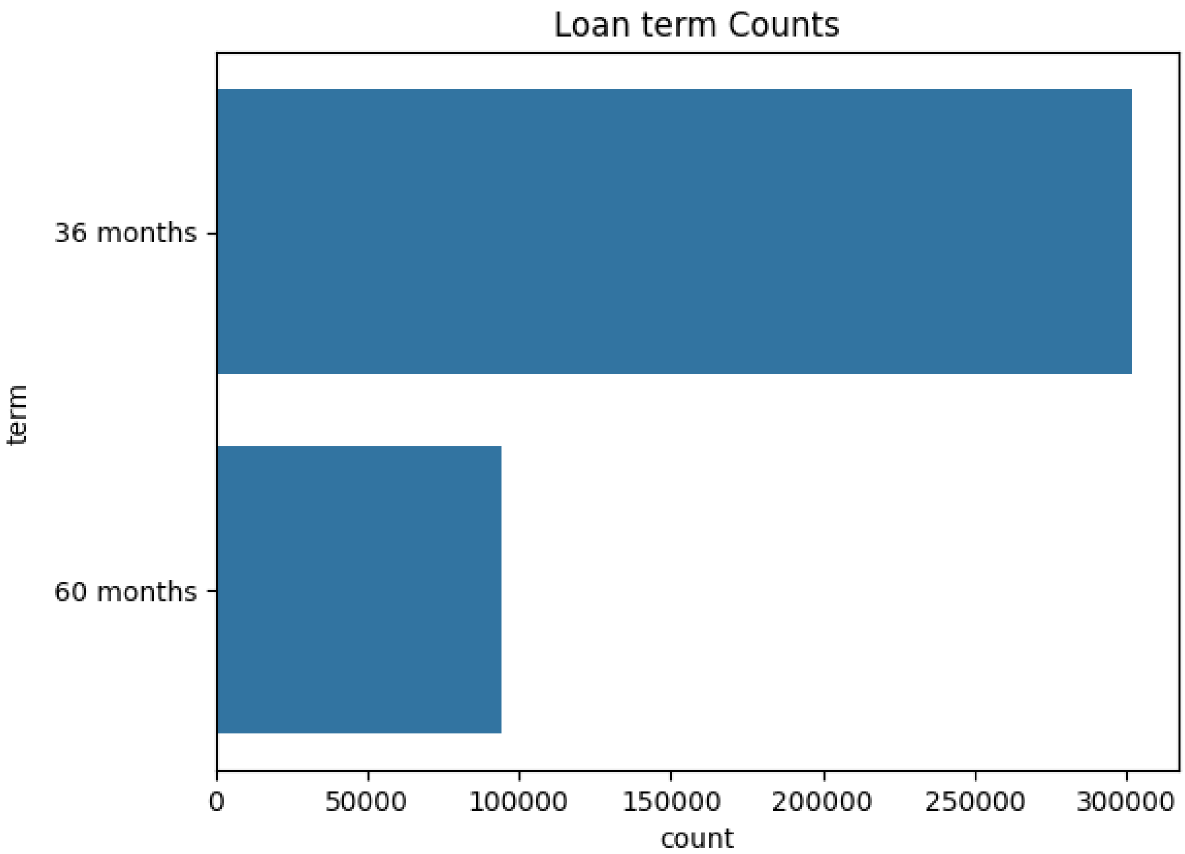


Sub-Grade and Their distribution
sub_grade
A1
A2
A3
A4
A5
B1
B2
B3
B4
B5
C1
C2
C3
C4
C5
D1
D2
D3
D4
D5
E1
E2
E3
E4
E5
F1
F2
F3
F4
F5



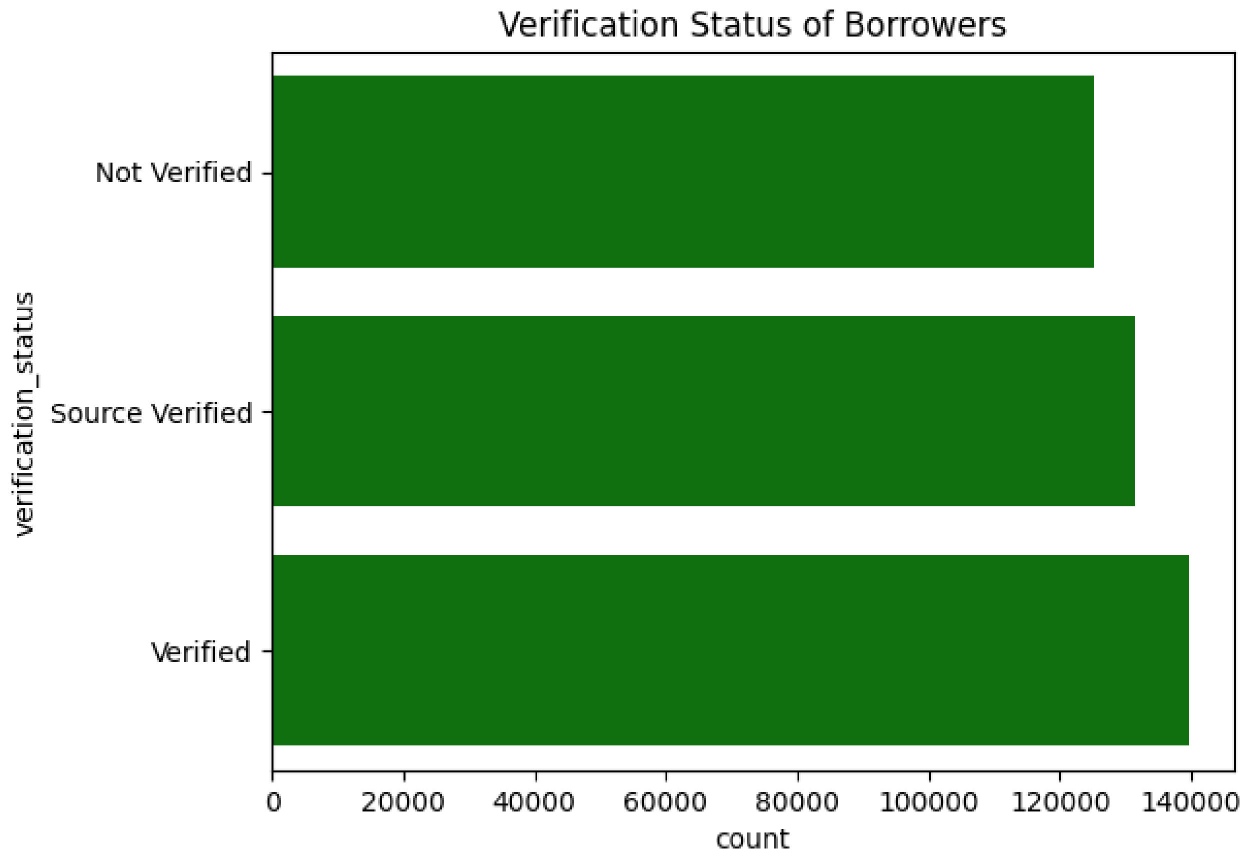
Grade and Their distribution
B
29%
C
27%
A
16%
D
16%
E
8%
F
3%
G
1%

```
sns.countplot(data['term'].sort_values())
plt.title('Loan term Counts')

plt.show()
```




```
sns.countplot(data['verification_status'].sort_values() , color='green')
plt.title('Verification Status of Borrowers')
```

```
plt.show()
```




```
print(data[categorical_data].describe())

                  term   grade sub_grade emp_title emp_length home_ownership  \
count           396030  396030    396030    373103     377729         396030
```

unique	2	7	35	173105	11	6
top	36 months	B	B3	Teacher	10+ years	MORTGAGE
freq	302005	116018	26655	4389	126041	198348

	verification_status	issue_d	loan_status	purpose \
count	396030	396030	396030	396030
unique	3	115	2	14
top	Verified	Oct-2014	Fully Paid	debt_consolidation
freq	139563	14846	318357	234507

	title	earliest_cr_line	initial_list_status \
count	394274	396030	396030
unique	48816	684	2
top	Debt consolidation	Oct-2000	f
freq	152472	3017	238066

	application_type	address
count	396030	396030
unique	3	393700
top	INDIVIDUAL	USCGC Smith\r\nFPO AE 70466
freq	395319	8

```
data[numerical_data].describe()
```

	loan_amnt	int_rate	installment	annual_inc \
count	396030.000000	396030.000000	396030.000000	3.960300e+05
mean	14113.888089	13.639400	431.849698	7.420318e+04
std	8357.441341	4.472157	250.727790	6.163762e+04
min	500.000000	5.320000	16.080000	0.000000e+00
25%	8000.000000	10.490000	250.330000	4.500000e+04
50%	12000.000000	13.330000	375.430000	6.400000e+04
75%	20000.000000	16.490000	567.300000	9.000000e+04
max	40000.000000	30.990000	1533.810000	8.706582e+06

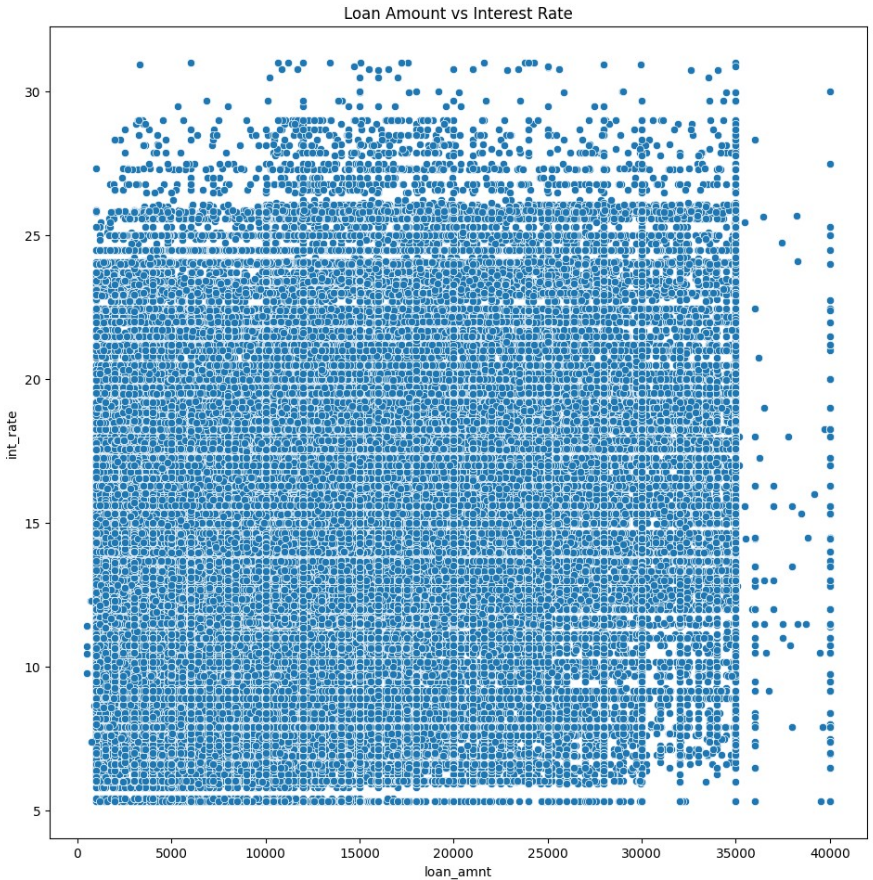
	dti	open_acc	pub_rec	revol_bal \

count	396030.000000	396030.000000	396030.000000	3.960300e+05
mean	17.379514	11.311153	0.178191	1.584454e+04
std	18.019092	5.137649	0.530671	2.059184e+04
min	0.000000	0.000000	0.000000	0.000000e+00
25%	11.280000	8.000000	0.000000	6.025000e+03
50%	16.910000	10.000000	0.000000	1.118100e+04
75%	22.980000	14.000000	0.000000	1.962000e+04
max	9999.000000	90.000000	86.000000	1.743266e+06

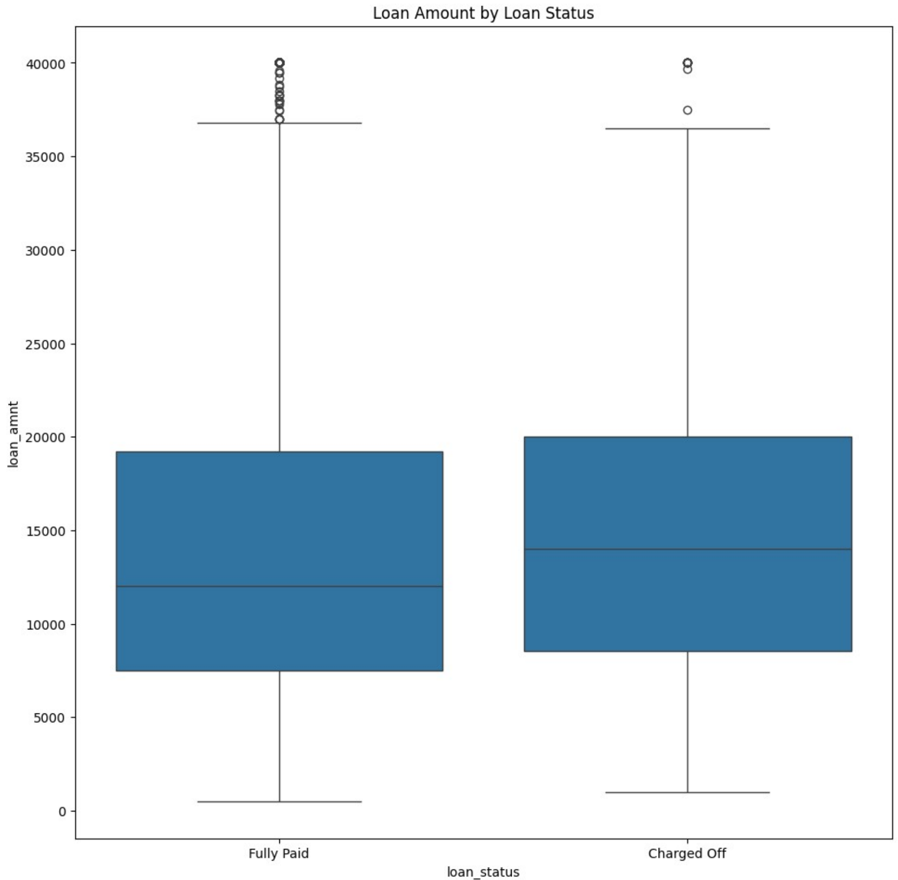
	revol_util	total_acc	mort_acc	pub_rec_bankruptcies
count	395754.000000	396030.000000	358235.000000	395495.000000
mean	53.791749	25.414744	1.813991	0.121648
std	24.452193	11.886991	2.147930	0.356174
min	0.000000	2.000000	0.000000	0.000000
25%	35.800000	17.000000	0.000000	0.000000
50%	54.800000	24.000000	1.000000	0.000000
75%	72.900000	32.000000	3.000000	0.000000
max	892.300000	151.000000	34.000000	8.000000

```python
plt.figure(figsize=(25,25))

plt.subplot(2,2,1)
# Scatter plot between loan amount and interest rate
sns.scatterplot(x='loan_amnt', y='int_rate', data=data)
plt.title('Loan Amount vs Interest Rate')

plt.subplot(2,2,2)
sns.boxplot(x=target_variable, y=data['loan_amnt'])
plt.title('Loan Amount by Loan Status')

plt.subplot(2,2,3)
sns.boxplot(x='grade', y='annual_inc', data=data, order=sorted(data['grade'].unique()))
plt.title('Annual Income by Loan Grade')

plt.subplot(2,2,4)
```

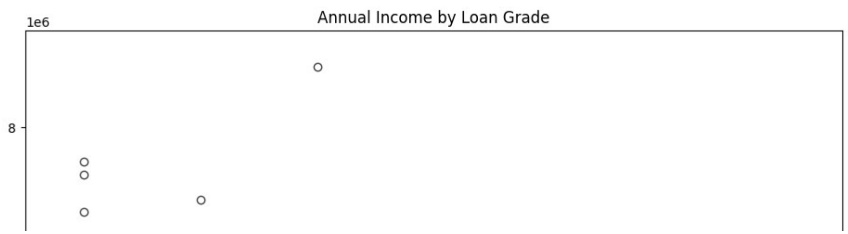
```
sns.boxplot(x='grade', y='loan_amnt', data=data)
plt.title('Loan Amount by Grade')



plt.show()
```



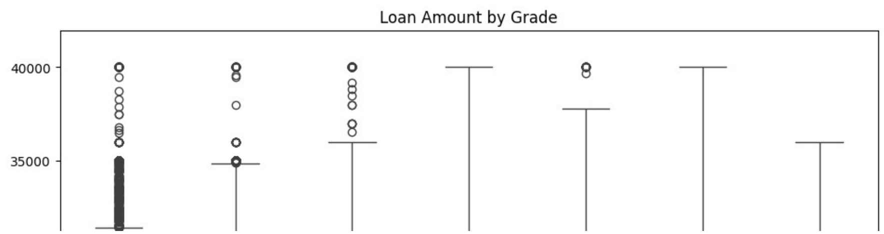
Loan Amount vs Interest Rate
int_rate
loan_amnt
0
5000
10000
15000
20000
25000
30000
35000
40000
5
10
15
20
25
30



Loan Amount by Loan Status
loan_amnt
loan_status
Fully Paid
Charged Off
0
5000
10000
15000
20000
25000
30000
35000
40000



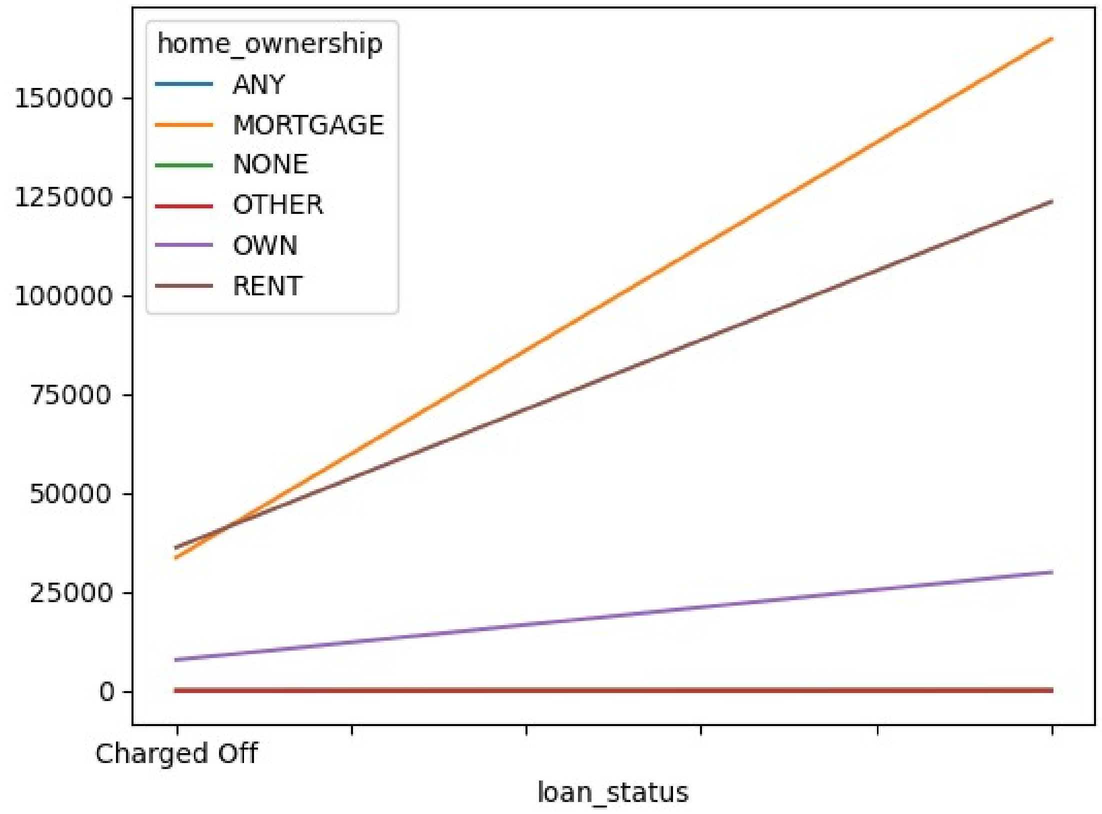
Annual Income by Loan Grade
1e6
8



Loan Amount by Grade
35000
40000

```python
crosstab = pd.crosstab(target_variable, data['home_ownership'])
print(crosstab)

crosstab.plot()
plt.show()
```

```
home_ownership  ANY  MORTGAGE  NONE  OTHER    OWN    RENT
loan_status
Charged Off       0     33632     7     16   7806   36212
Fully Paid        3    164716    24     96  29940  123578
```



Comments:

`a. On range of attributes`

- Loan amount ranges from 500-40000
- Range of Interest Rate : 5.3% - 31%
- Average annual Income : 74203
- Average Debt to Income Ratio is : 7.4%
- Average number of open credit lines in the borrower's credit file : 11.3

- Total Deregatory Records : 396030
- Average Revolving Balance : 15844.5

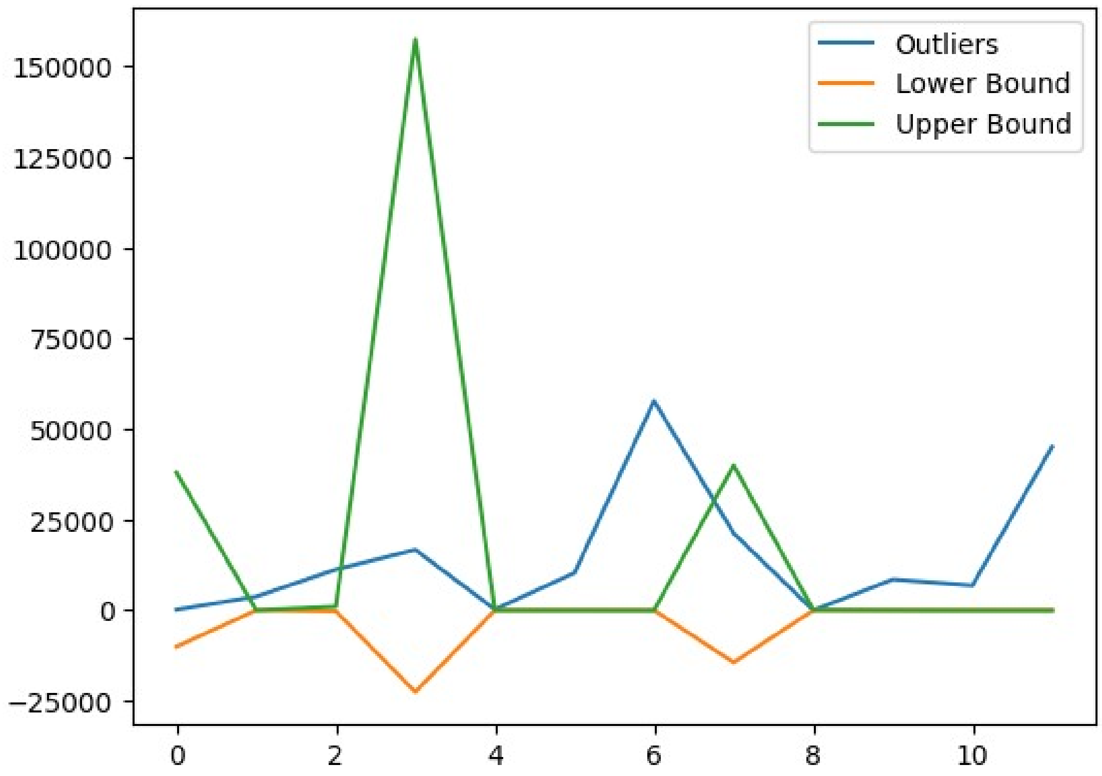
```
data['revol_bal'].mean()
```

```
15844.539853041437
```

```
b. Outliers of various attributes
```

```
# funcion to calculate lower bound & upper bound of a feature
def detect_outliers_iqr(column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = data[(data[column] < lower_bound) | (data[column] > upper_bound)]
    return outliers, lower_bound,upper_bound

outliers_summary_list = []

# Loop through numerical columns and calculate outliers
for col in numerical_data:
    outliers, lower, upper = detect_outliers_iqr(col)

    # Append the results as a dictionary to the list
    outliers_summary_list.append({
        'Column': col,
        'Outliers': len(outliers),
        'Lower Bound': lower,
        'Upper Bound': upper
    })

# Convert the list of dictionaries into a DataFrame
outliers_summary = pd.DataFrame(outliers_summary_list)
```

```python
# Display the summary DataFrame
outliers_summary.plot()
plt.show()
```



2. Data Preprocessing

a. Duplicate value check

```python
print(f"Number of duplicate rows: {data[data.duplicated()].shape[0]}")
```

```
Number of duplicate rows: 0
```

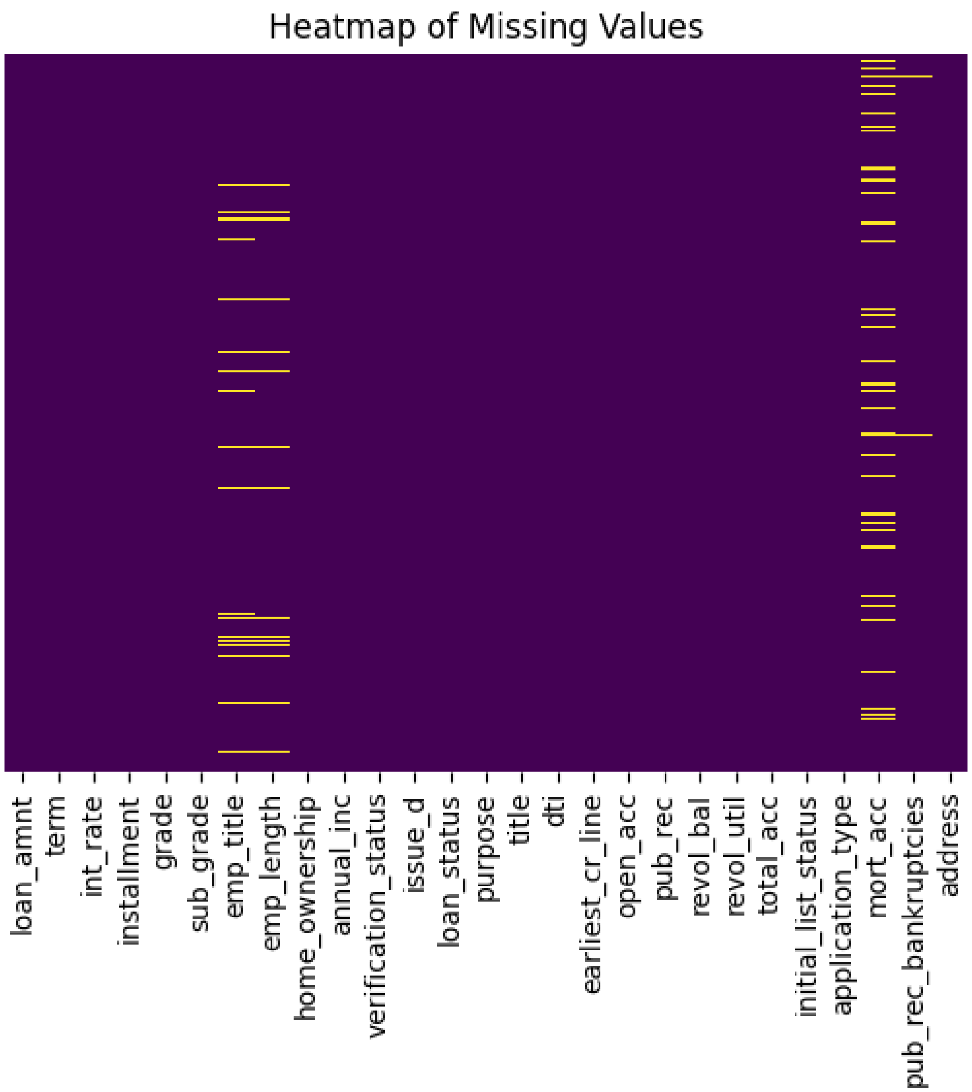
```python
data.nunique()
```

```
loan_amnt                 1397
term                         2
int_rate                   566
installment              55706
grade                        7
sub_grade                   35
emp_title               173105
emp_length                  11
home_ownership               6
annual_inc               27197
verification_status          3
issue_d                    115
loan_status                  2
purpose                     14
title                    48816
dti                       4262
earliest_cr_line           684
open_acc                    61
pub_rec                     20
revol_bal                55622
revol_util                1226
total_acc                  118
initial_list_status          2
application_type             3
mort_acc                    33
```

```
pub_rec_bankruptcies          9
address                  393700
dtype: int64
```

b. Missing value treatment

```
# Identify Missing Values
missing_values_data = data.isnull().sum()
print( missing_values_data[missing_values_data > 0])

emp_title               22927
emp_length              18301
title                    1756
revol_util                276
mort_acc                37795
pub_rec_bankruptcies      535
dtype: int64

sns.heatmap(data.isnull(), cbar=False, cmap='viridis', yticklabels=False)
plt.title('Heatmap of Missing Values')
plt.show()
```


Heatmap of Missing Values
loan_amnt
term
int_rate
installment
grade
sub_grade
emp_title
emp_length
home_ownership
annual_inc
verification_status
issue_d
loan_status
purpose
title
dti
earliest_cr_line
open_acc
pub_rec
revol_bal
revol_util
total_acc
initial_list_status
application_type
mort_acc
pub_rec_bankruptcies
address

```python
# Calculating the percentage of missing values
missing_percentage = round((missing_values_data / len(data)) * 100,2)

missing_data_summary = pd.DataFrame({
    'Missing Values': missing_values_data[missing_values_data > 0],
    'Percentage (%)': missing_percentage[missing_values_data > 0]
}).sort_values(by='Percentage (%)', ascending=False)

missing_data_summary
```

	Missing Values	Percentage (%)
mort_acc	37795	9.54
emp_title	22927	5.79
emp_length	18301	4.62
title	1756	0.44
pub_rec_bankruptcies	535	0.14
revol_util	276	0.07

Treating Categorical Missing values

- Replace missing values with "Unknown" or "Not Provided".
- Alternatively, drop the column if it’s not critical to the analysis.

```
 Categorical data 'Job titles' might be critical to the analysis

data['emp_title']=data['emp_title'].fillna('Unknown')

 Categorical data 'Employment duration' critical to the analysis

data['emp_length']=data['emp_length'].fillna('Not Provided')

data['emp_length'].isna().sum()

0
```

Treating `title` by replacing missing values with the mode as very less percentage is missing

Can also be droped, Not critical to the analysis

```
data['title']=data['title'].fillna(data['title'].mode()[0])
```

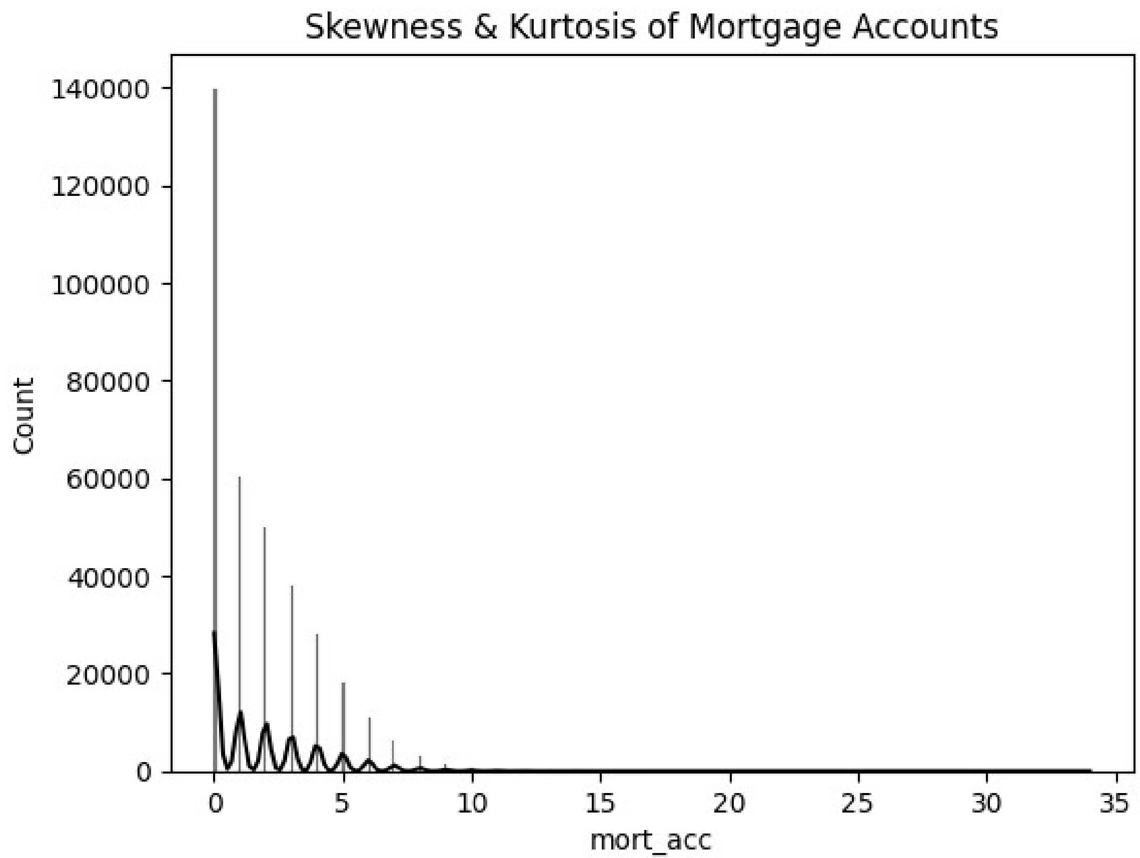
Handling Numerical missing data

mort_acc :

- Replace with the median or mode, as it's a small percentage and likely skewed

```
# Calculate skewness and kurtosis
print("Skewness:", data['mort_acc'].skew())
print("Kurtosis:", data['mort_acc'].kurt())

sns.histplot(data=data['mort_acc'],kde=True, color='black')
plt.title('Skewness & Kurtosis of Mortgage Accounts')
plt.show()
```

```
Skewness: 1.6001324380874855
Kurtosis: 4.477175725939146
```




```
data['mort_acc']=data['mort_acc'].fillna(data['mort_acc'].median())
```

Treating pub_rec_bankruptcies by replacing missing values with the mode

```
data['pub_rec_bankruptcies']=data['pub_rec_bankruptcies'].fillna(data['pub_rec_bankruptcies'].mode()[0])
```

Since revol_util : the missing percentage is small, filling with the median is straightforward and retains the dataset's size.

```
data['revol_util']=data['revol_util'].fillna(data['revol_util'].median())

data.isnull().sum().sum()

0

 Zero missing values remain
```

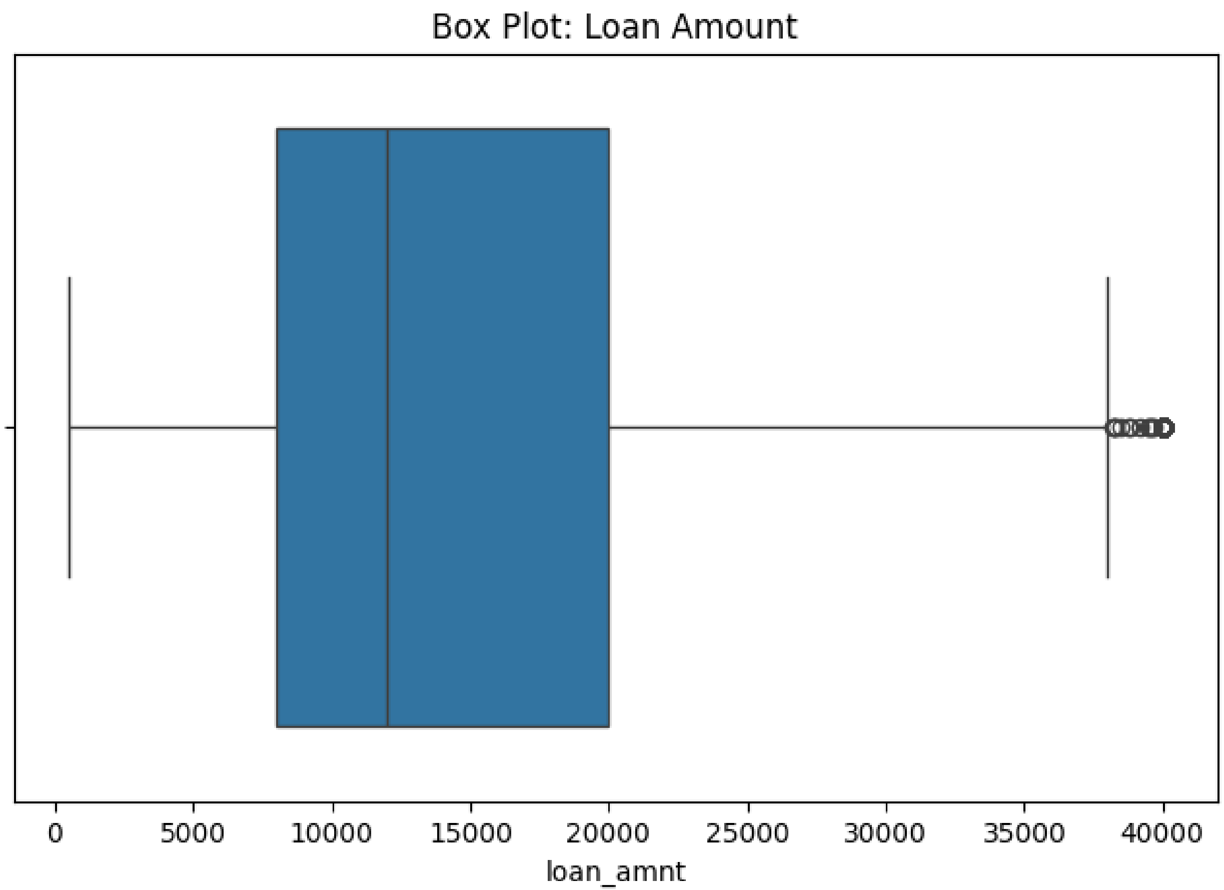
---

c. Outlier Treatment

```
 detect_outliers_iqr function is already defined in the abouve cells to reduce code redundancy
```

Loan amount 'loan_amnt'

```
# Extraction upper,lower bound & outlier
loan_amnt_outliers, loan_amnt_lower, loan_amnt_upper = detect_outliers_iqr('loan_amnt')

plt.figure(figsize=(8, 5))
sns.boxplot(x=data['loan_amnt'])
plt.title('Box Plot: Loan Amount')
plt.show()
```

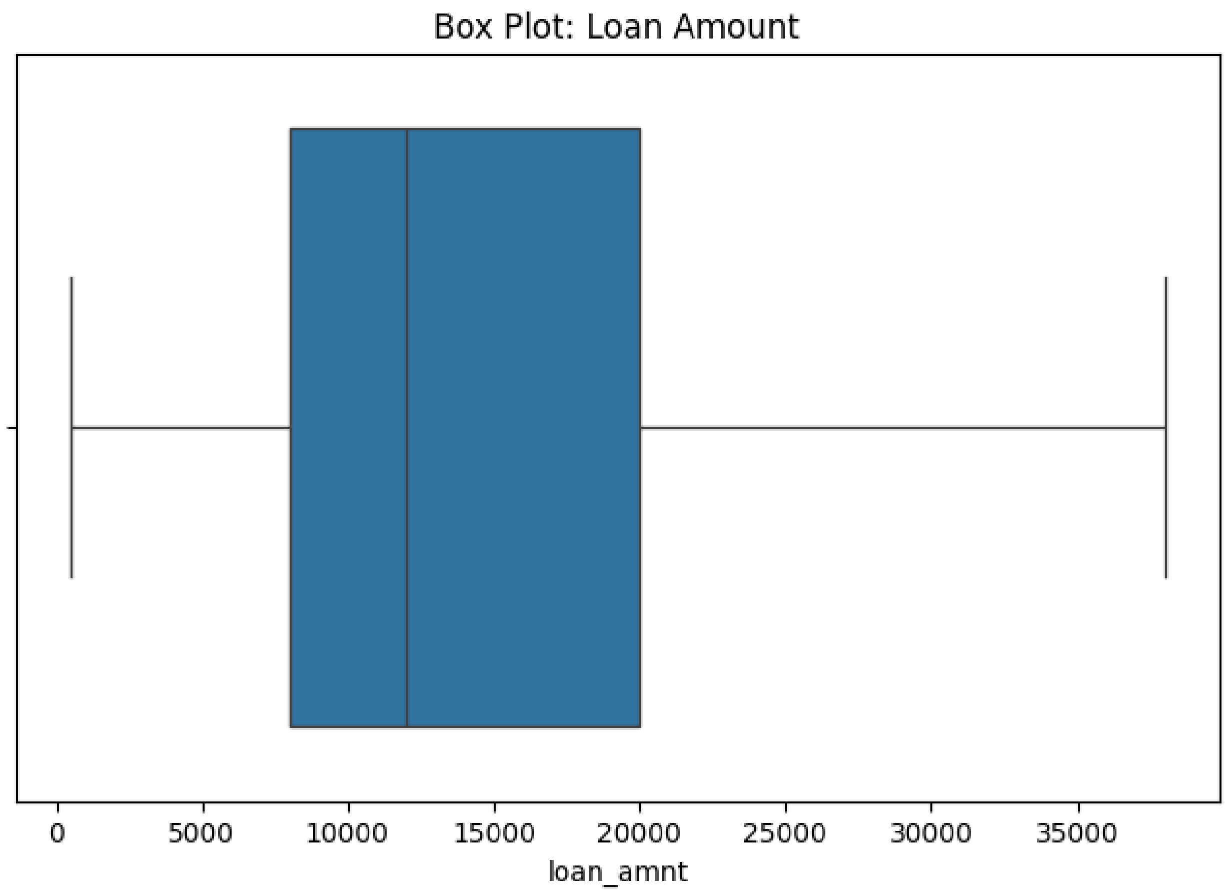



```
len(loan_amnt_outliers), loan_amnt_lower, loan_amnt_upper
```

```
(191, -10000.0, 38000.0)
```

as we can see their are 191 outliers in this feature we will be using caping

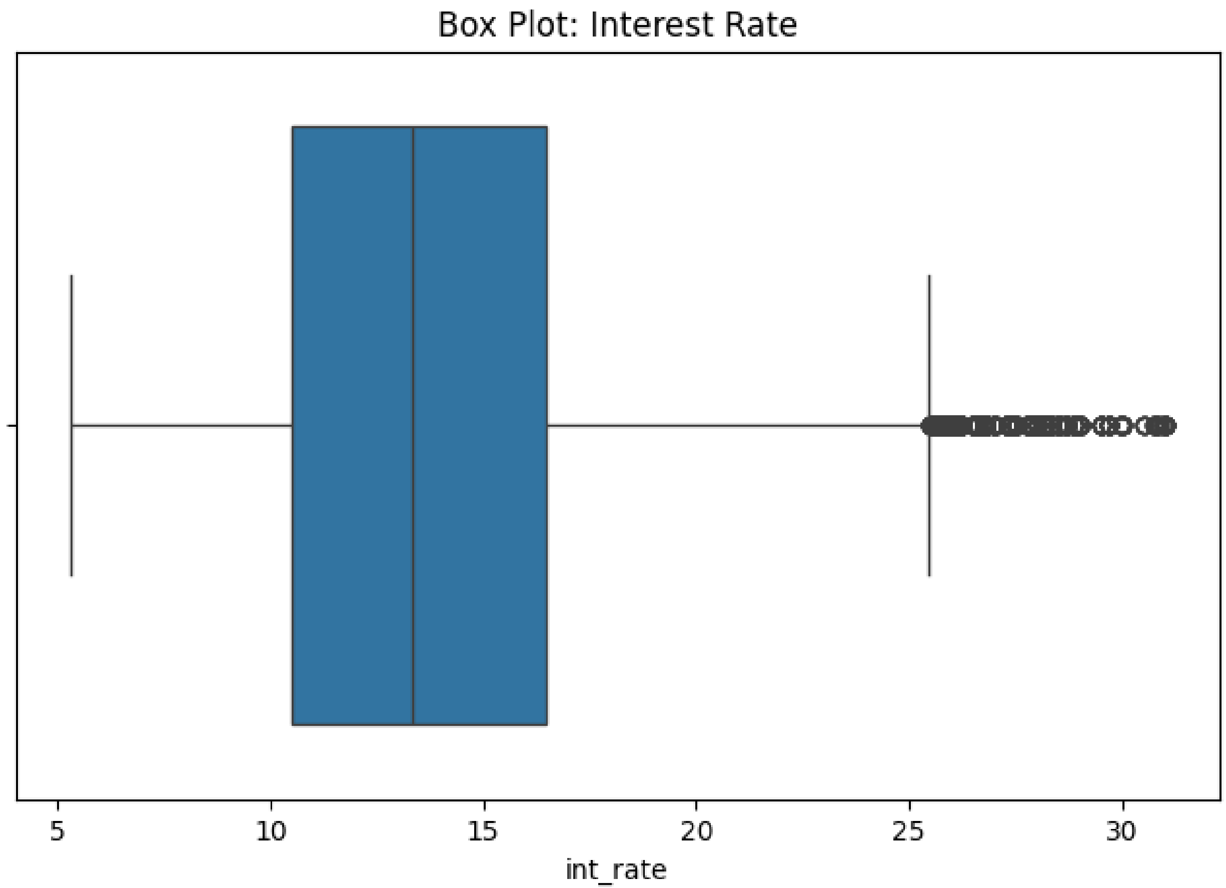
```python
# Cap the outliers
data['loan_amnt'] = np.where(data['loan_amnt'] < loan_amnt_lower, loan_amnt_lower, data['loan_amnt'])
data['loan_amnt'] = np.where(data['loan_amnt'] > loan_amnt_upper, loan_amnt_upper, data['loan_amnt'])

plt.figure(figsize=(8, 5))
sns.boxplot(x=data['loan_amnt'])
plt.title('Box Plot: Loan Amount')
plt.show()
```



Interest Rate 'int_rate'

```
# Extraction upper,lower bound & outlier
int_rate_outliers, int_rate_lower, int_rate_upper = detect_outliers_iqr('int_rate')
```

```
plt.figure(figsize=(8, 5))
sns.boxplot(x=data['int_rate'])
plt.title('Box Plot: Interest Rate')
plt.show()

len(int_rate_outliers), int_rate_lower, int_rate_upper
```



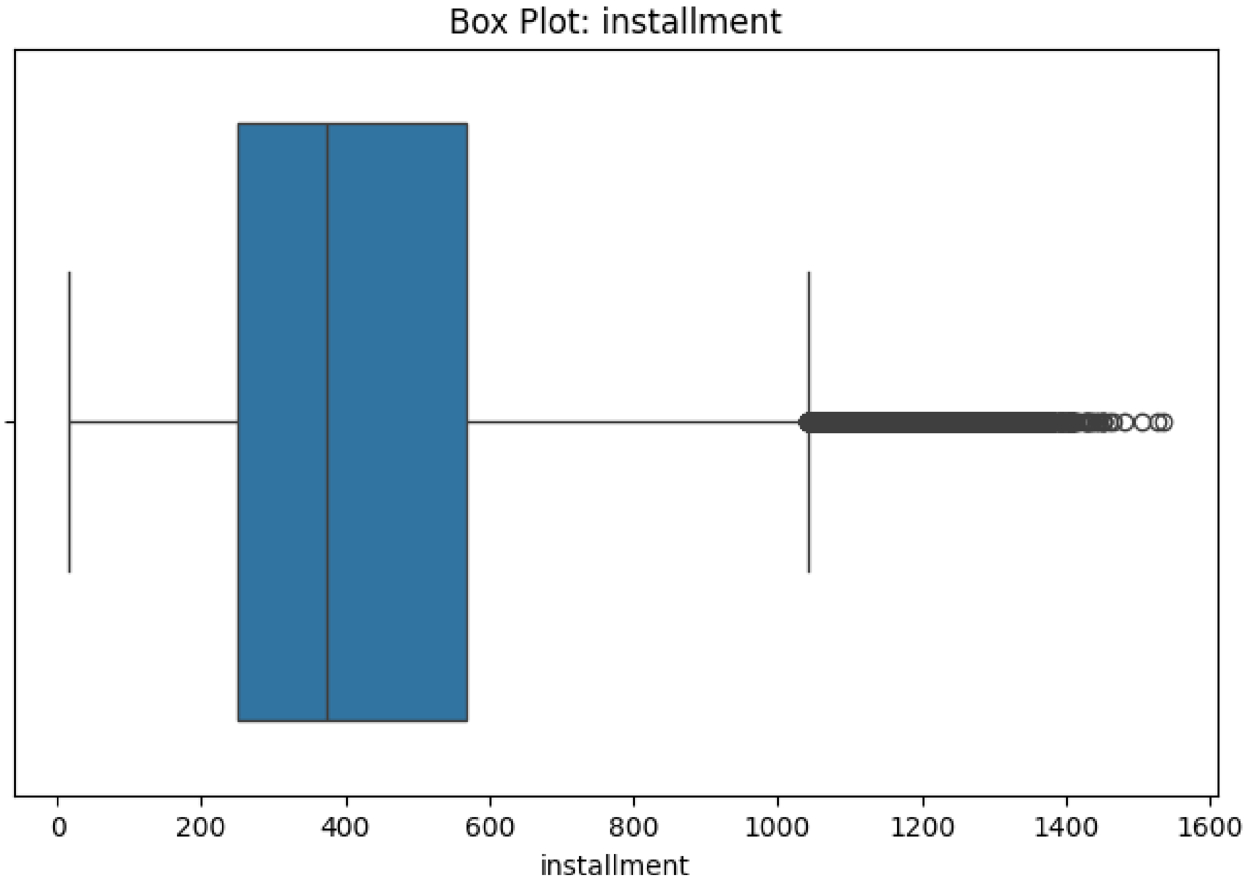
```
(3777, 1.4900000000000038, 25.489999999999995)
```

By Domain Knowledge we know The inerest rate is critical to the analysis,Hence no treatment should be performed

---

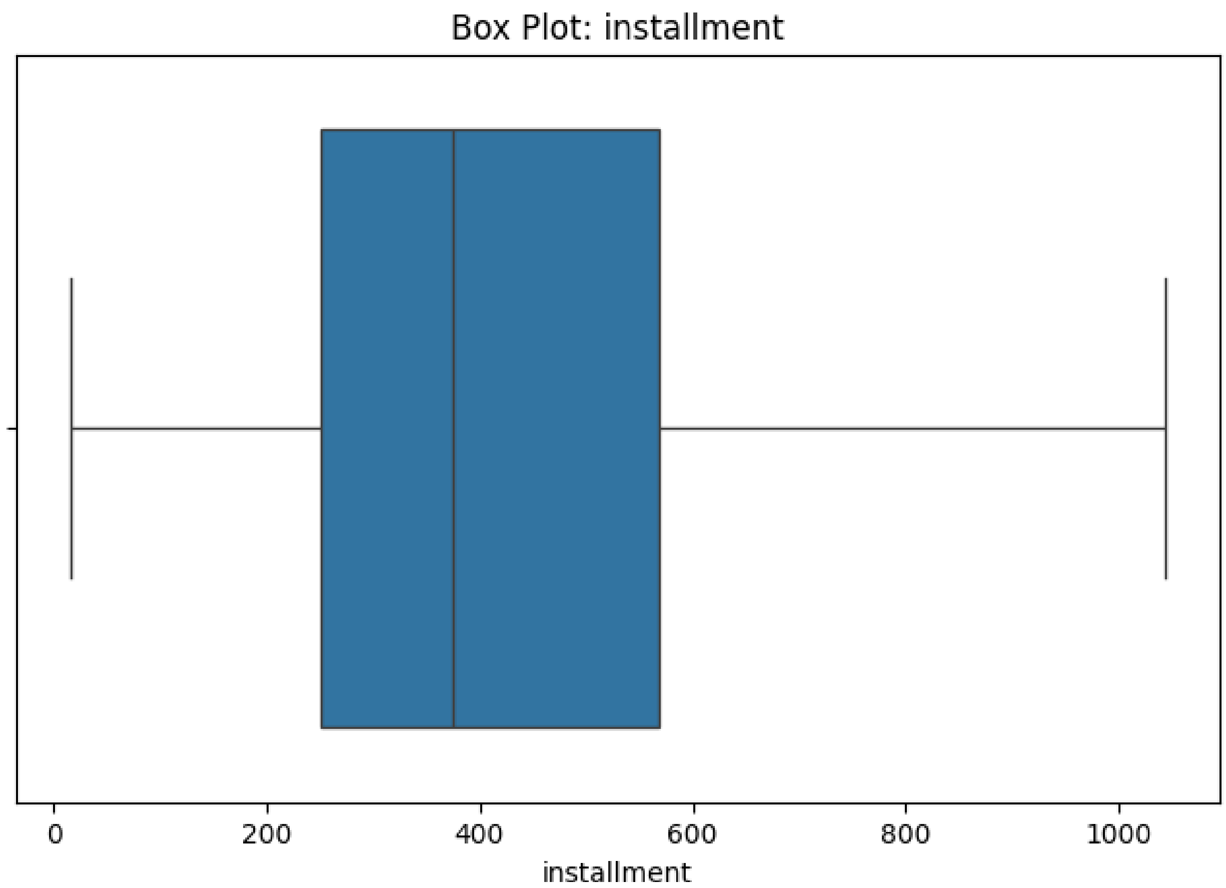
installment

```
# Extraction upper,lower bound & outlier
installment_outliers, installment_lower, installment_upper = detect_outliers_iqr('installment')

plt.figure(figsize=(8, 5))
sns.boxplot(x=data['installment'])
plt.title('Box Plot: installment')
plt.show()

len(installment_outliers), installment_lower, installment_upper
```




```
(11250, -225.12499999999986, 1042.7549999999999)

# Cap the outliers
data['installment'] = np.where(data['installment'] < installment_lower, installment_lower,
data['installment'])
data['installment'] = np.where(data['installment'] > installment_upper, installment_upper,
data['installment'])
```

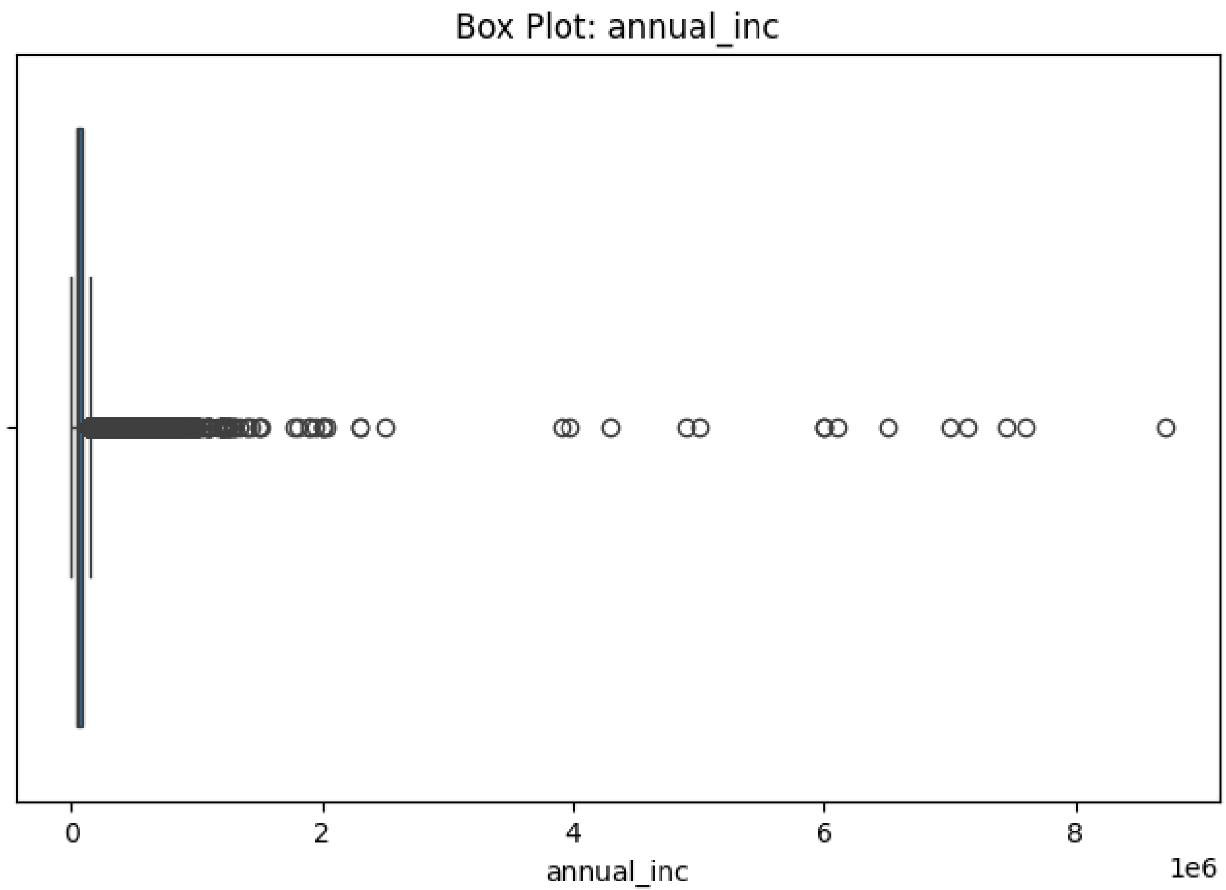
```
plt.figure(figsize=(8, 5))
sns.boxplot(x=data['installment'])
plt.title('Box Plot: installment')
plt.show()
```



Annual income 'annual_inc'

no treatment should be done ,data critical to analysis

```
# Extraction upper,lower bound & outlier
annual_inc_outliers, annual_inc_lower, annual_inc_upper = detect_outliers_iqr('annual_inc')

plt.figure(figsize=(8, 5))
sns.boxplot(x=data['annual_inc'])
plt.title('Box Plot: annual_inc')
plt.show()

len(annual_inc_outliers), annual_inc_lower, annual_inc_upper
```

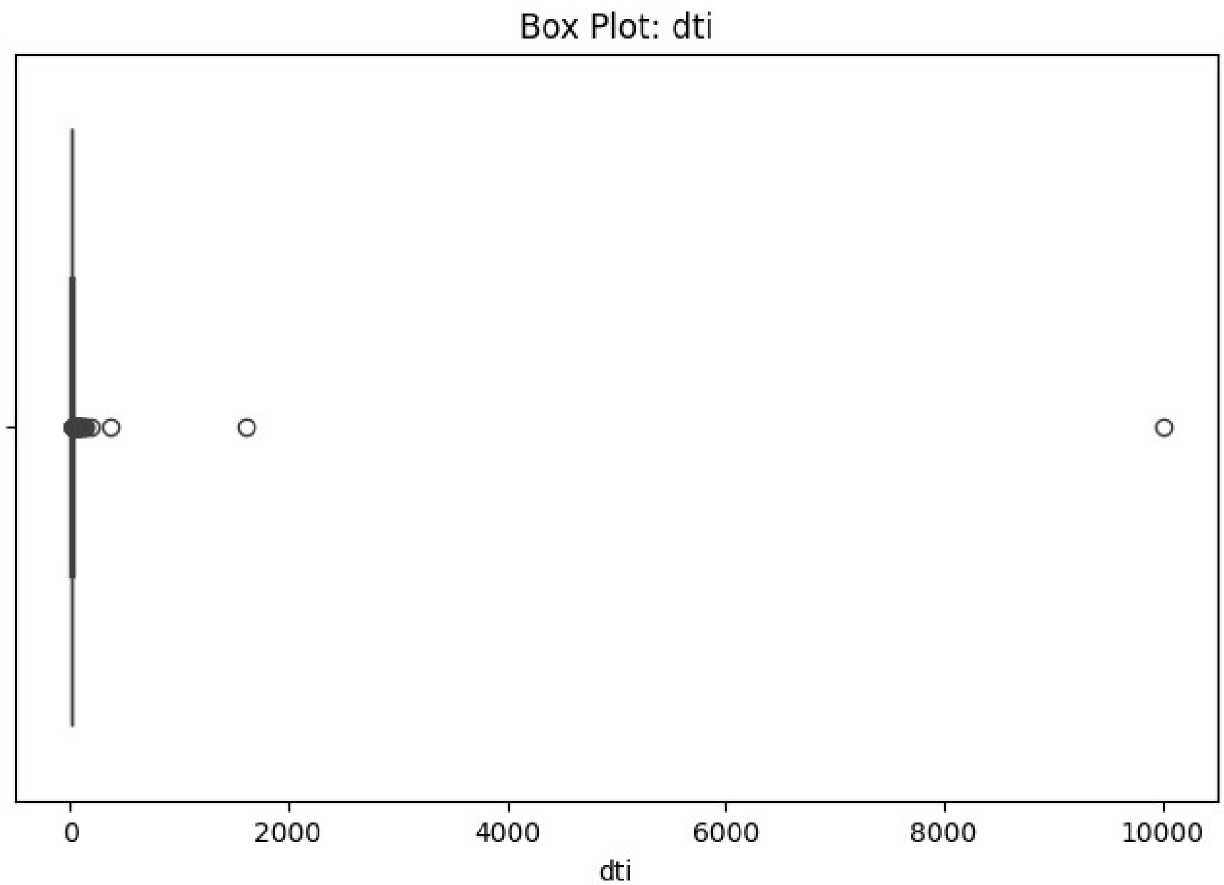



```
(16700, -22500.0, 157500.0)
```

Debt to Income ratio 'dti'

This is also can not be treated as it has much information

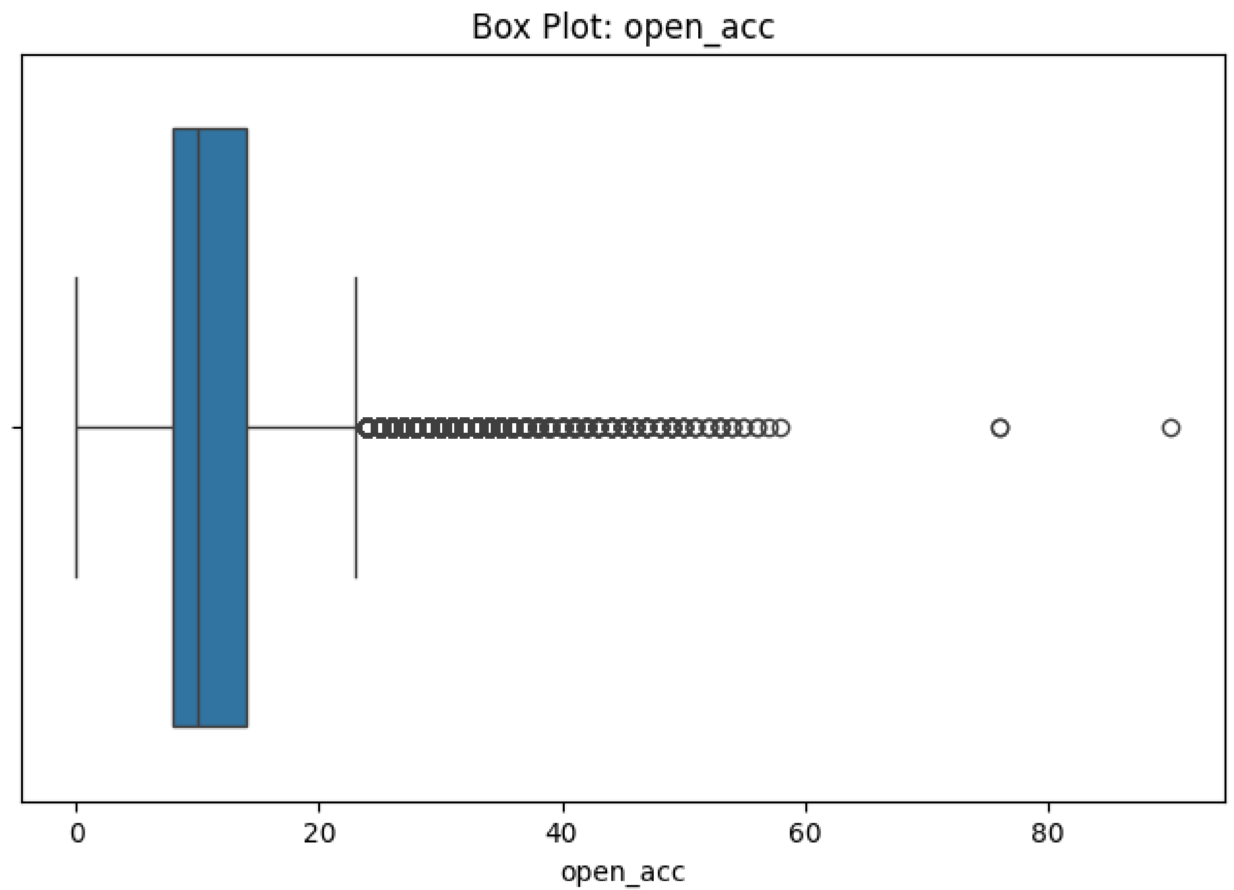
```python
# Extraction upper,lower bound & outlier
dti_outliers, dti_lower, dti_upper = detect_outliers_iqr('dti')

plt.figure(figsize=(8, 5))
sns.boxplot(x=data['dti'])
plt.title('Box Plot: dti')
plt.show()

len(dti_outliers), dti_lower, dti_upper
```

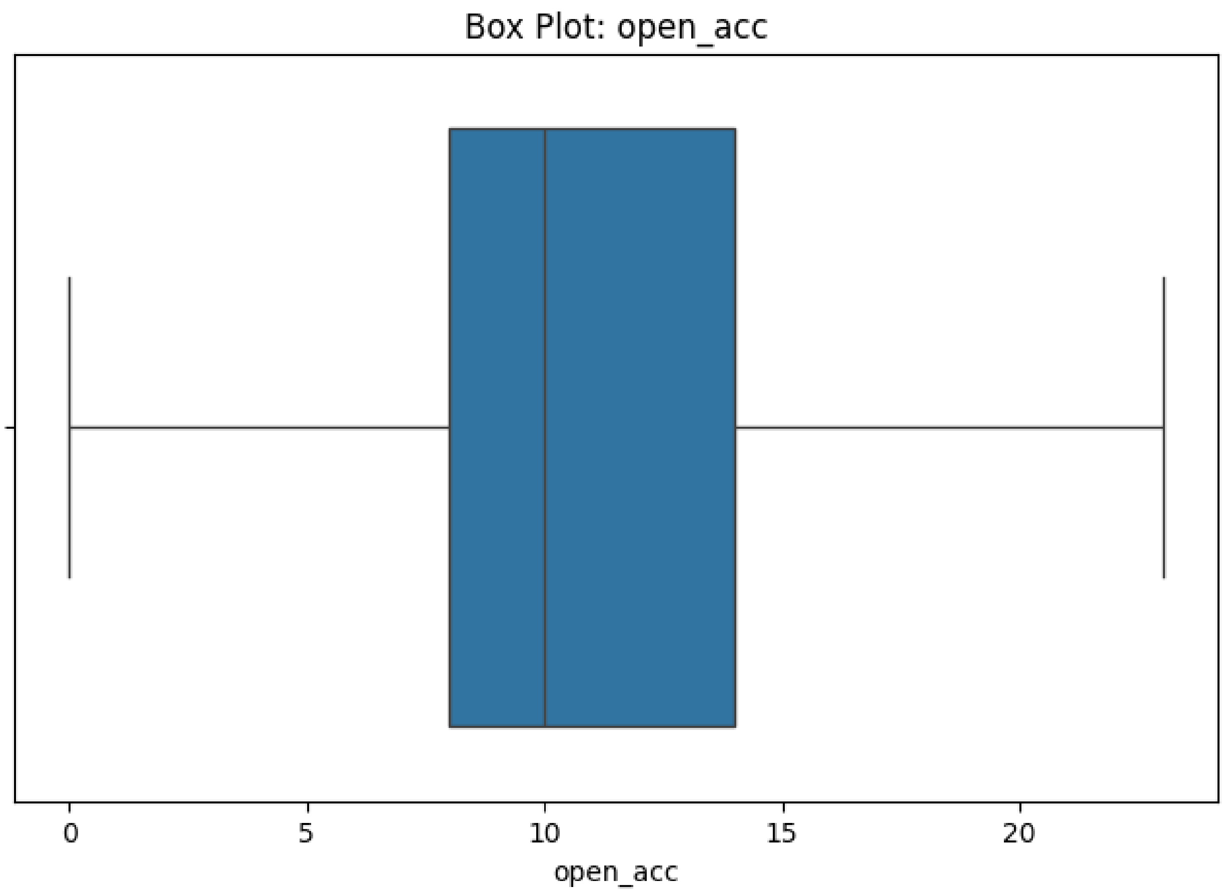



```
(275, -6.270000000000001, 40.53)
```

Open account 'open_acc'

```python
# Extraction upper,lower bound & outlier
open_acc_outliers, open_acc_lower, open_acc_upper = detect_outliers_iqr('open_acc')

plt.figure(figsize=(8, 5))
sns.boxplot(x=data['open_acc'])
plt.title('Box Plot: open_acc')
plt.show()

len(open_acc_outliers), open_acc_lower, open_acc_upper

# Cap the outliers
data['open_acc'] = np.where(data['open_acc'] < open_acc_lower, open_acc_lower, data['open_acc'])
data['open_acc'] = np.where(data['open_acc'] > open_acc_upper, open_acc_upper, data['open_acc'])

plt.figure(figsize=(8, 5))
sns.boxplot(x=data['open_acc'])
plt.title('Box Plot: open_acc')
plt.show()
```

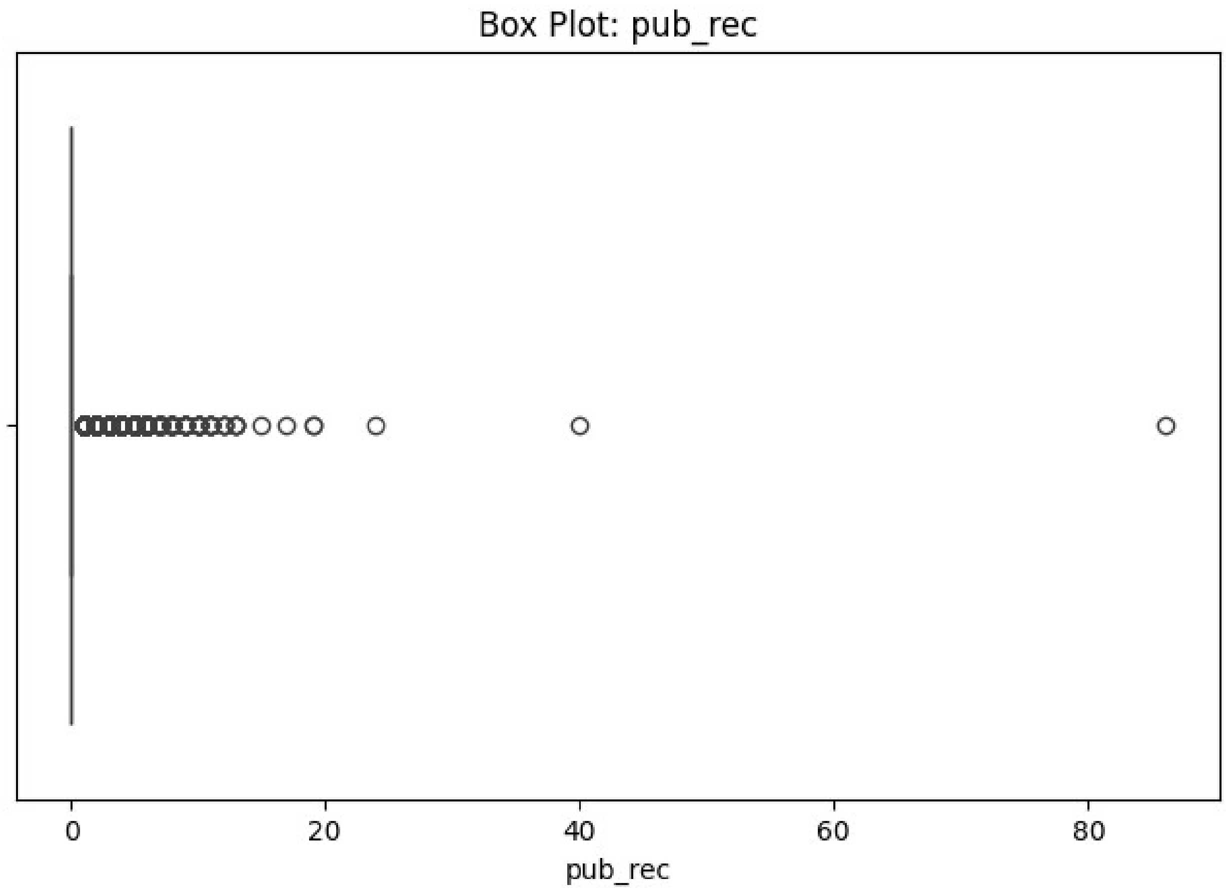

Box Plot: open_acc
0
20
40
60
80
open_acc



Public Recedings 'pub_rec'

```
# Extraction upper,lower bound & outlier
pub_rec_outliers, pub_rec_lower, pub_rec_upper = detect_outliers_iqr('pub_rec')
```

```
plt.figure(figsize=(8, 5))
sns.boxplot(x=data['pub_rec'])
plt.title('Box Plot: pub_rec')
plt.show()

len(pub_rec_outliers), pub_rec_lower, pub_rec_upper
```

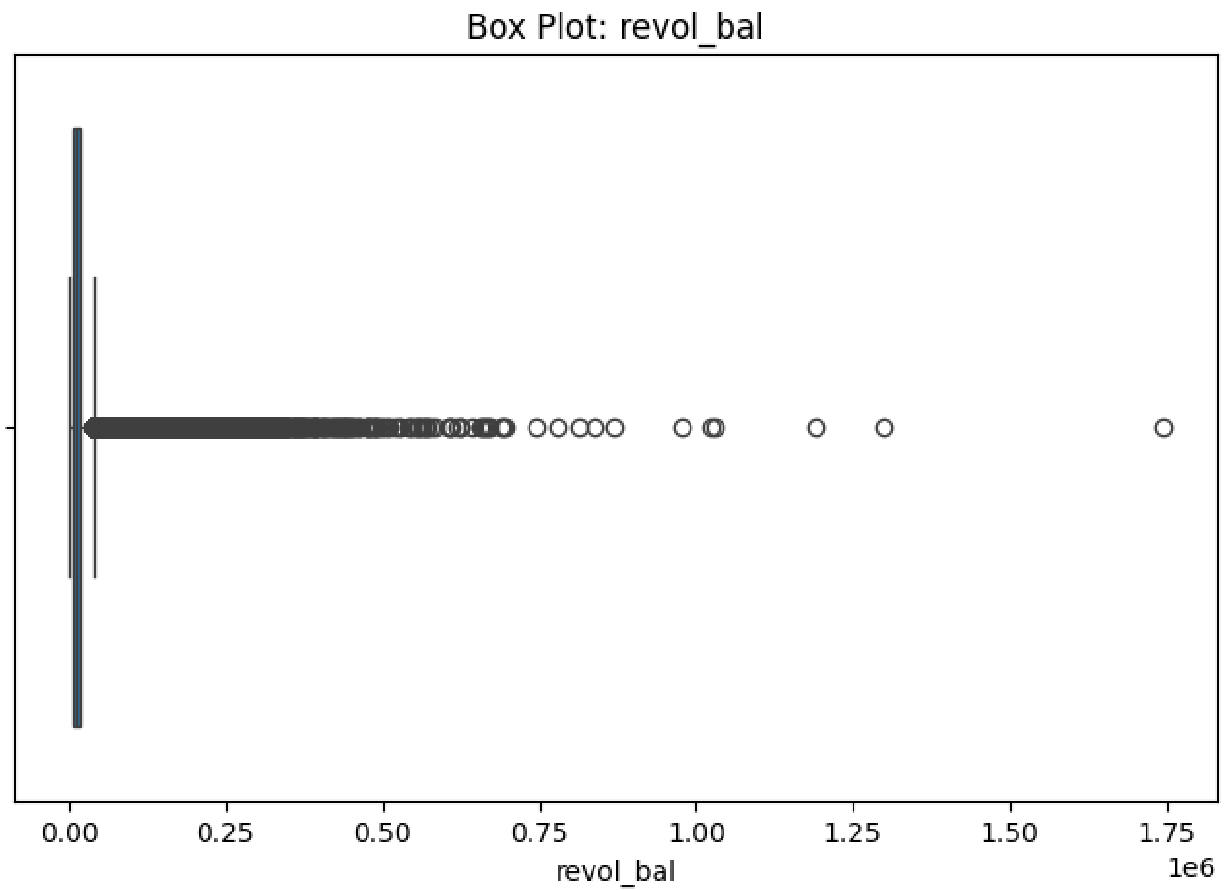


```
(57758, 0.0, 0.0)
```

critical to the analysis , should not be treated

Revolving Balance 'revol_bal'

```
# Extraction upper,lower bound & outlier
revol_bal_outliers, revol_bal_lower, revol_bal_upper = detect_outliers_iqr('revol_bal')

plt.figure(figsize=(8, 5))
sns.boxplot(x=data['revol_bal'])
plt.title('Box Plot: revol_bal')
plt.show()

len(revol_bal_outliers), revol_bal_lower, revol_bal_upper
```

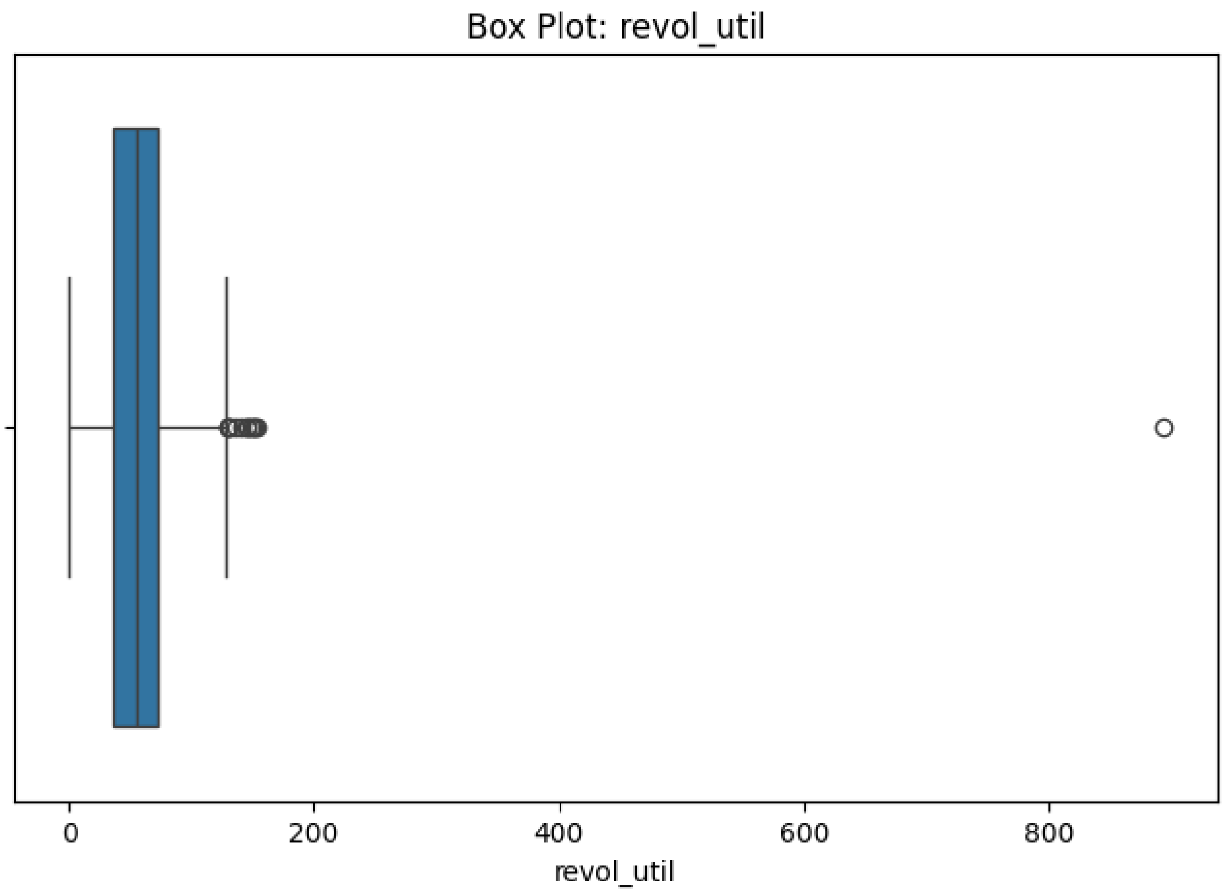



```
(21259, -14367.5, 40012.5)
```

can not be treated

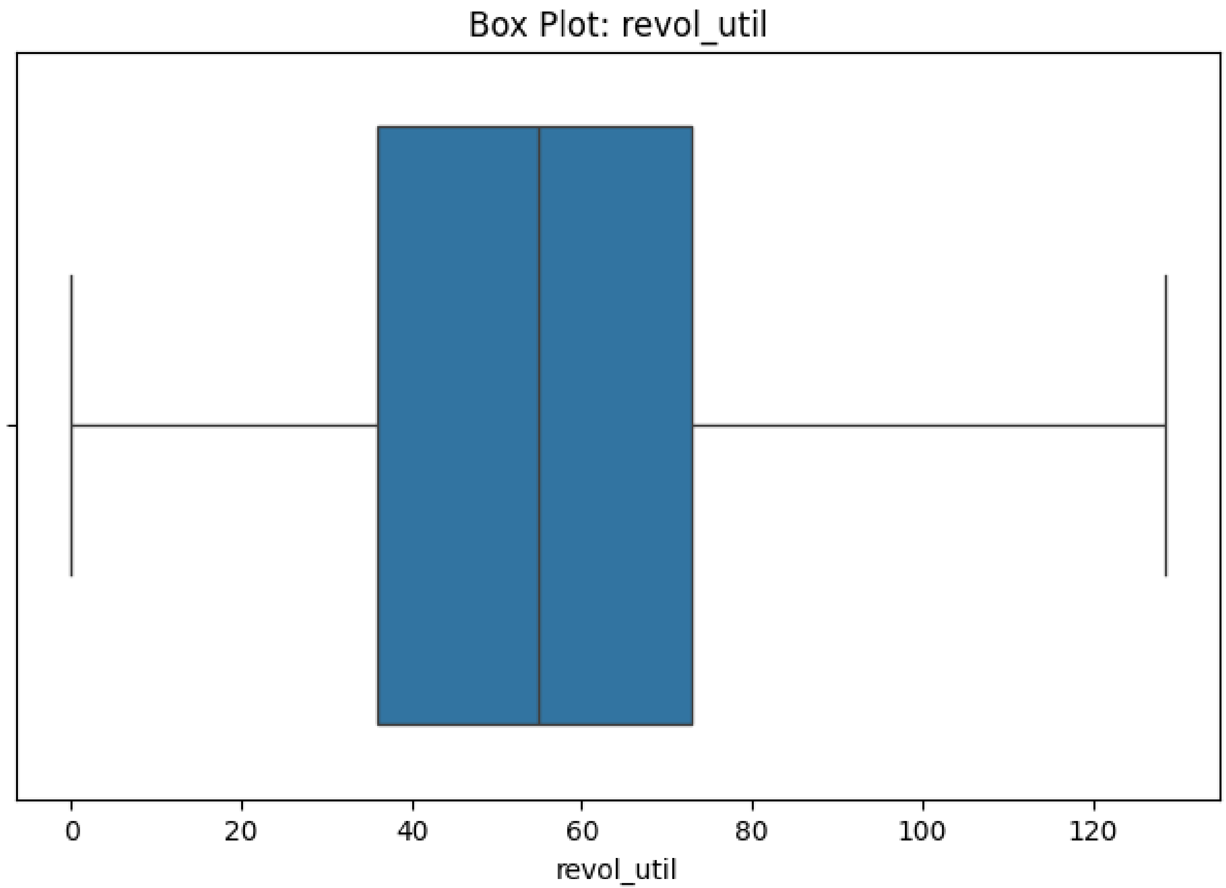
Revolving Util 'revol_util'

```python
# Extraction upper,lower bound & outlier
revol_util_outliers, revol_util_lower, revol_util_upper = detect_outliers_iqr('revol_util')

plt.figure(figsize=(8, 5))
sns.boxplot(x=data['revol_util'])
plt.title('Box Plot: revol_util')
plt.show()

len(revol_util_outliers), revol_util_lower, revol_util_upper
```



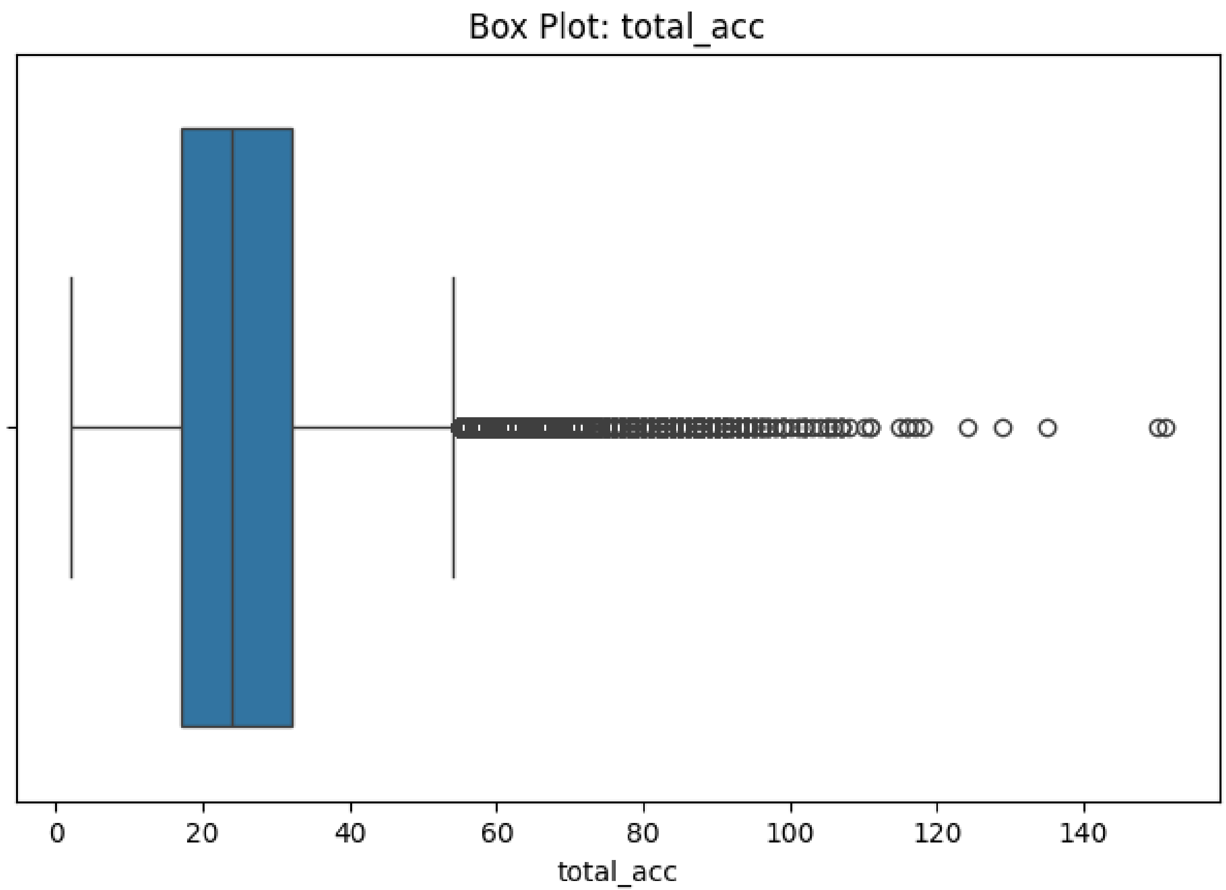

```
(12, -19.600000000000016, 128.40000000000003)

# Cap the outliers
data['revol_util'] = np.where(data['revol_util'] < revol_util_lower, revol_util_lower,
data['revol_util'])
data['revol_util'] = np.where(data['revol_util'] > revol_util_upper, revol_util_upper,
data['revol_util'])
```

```
plt.figure(figsize=(8, 5))
sns.boxplot(x=data['revol_util'])
plt.title('Box Plot: revol_util')
plt.show()
```



Total Accounts 'total_acc'

```
# Extraction upper,lower bound & outlier
total_acc_outliers, total_acc_lower, total_acc_upper = detect_outliers_iqr('total_acc')

plt.figure(figsize=(8, 5))
sns.boxplot(x=data['total_acc'])
plt.title('Box Plot: total_acc')
plt.show()

len(total_acc_outliers), total_acc_lower, total_acc_upper
```

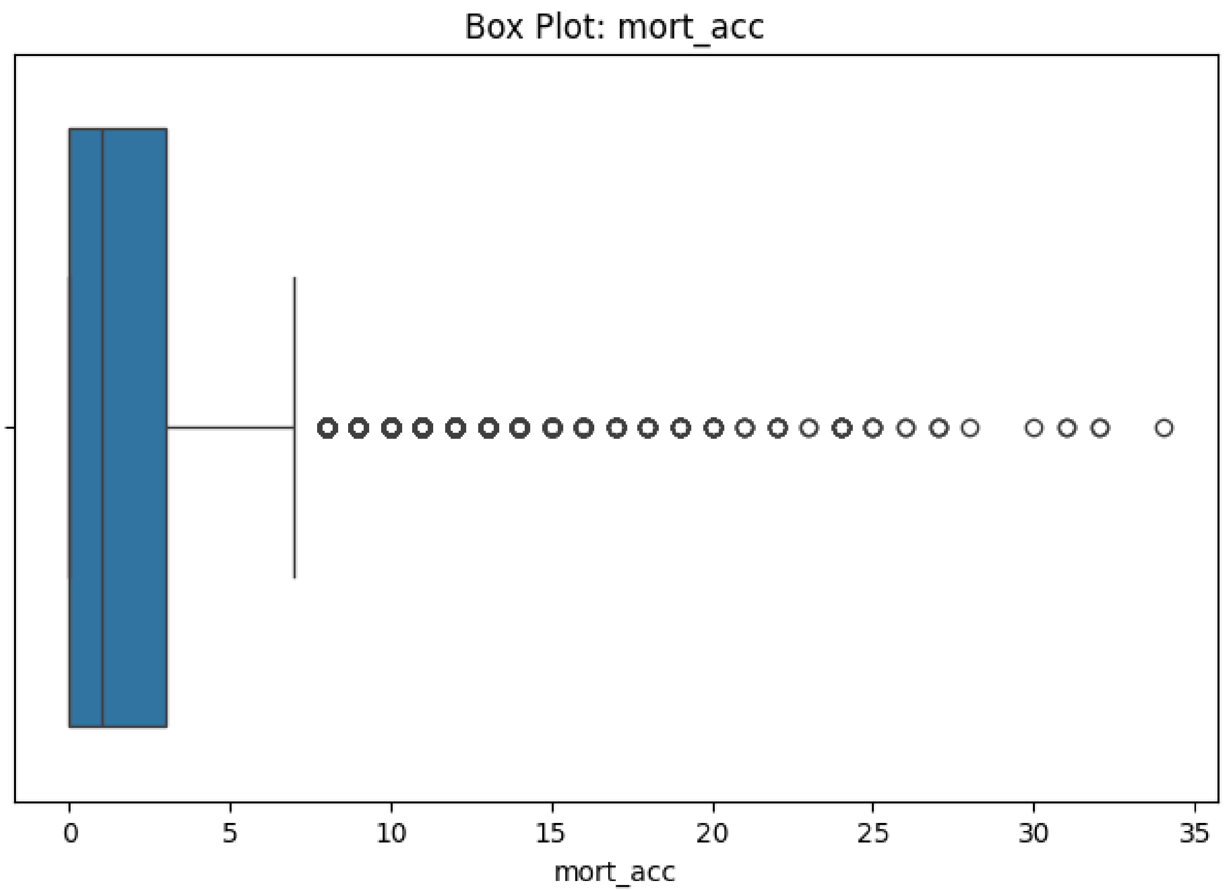



```
(8499, -5.5, 54.5)
```

has valuable informatoin, should not be treated

Mortgage Accounts 'mort_acc'

```python
# Extraction upper,lower bound & outlier
mort_acc_outliers, mort_acc_lower, mort_acc_upper = detect_outliers_iqr('mort_acc')

plt.figure(figsize=(8, 5))
sns.boxplot(x=data['mort_acc'])
plt.title('Box Plot: mort_acc')
plt.show()

len(mort_acc_outliers), mort_acc_lower, mort_acc_upper
```



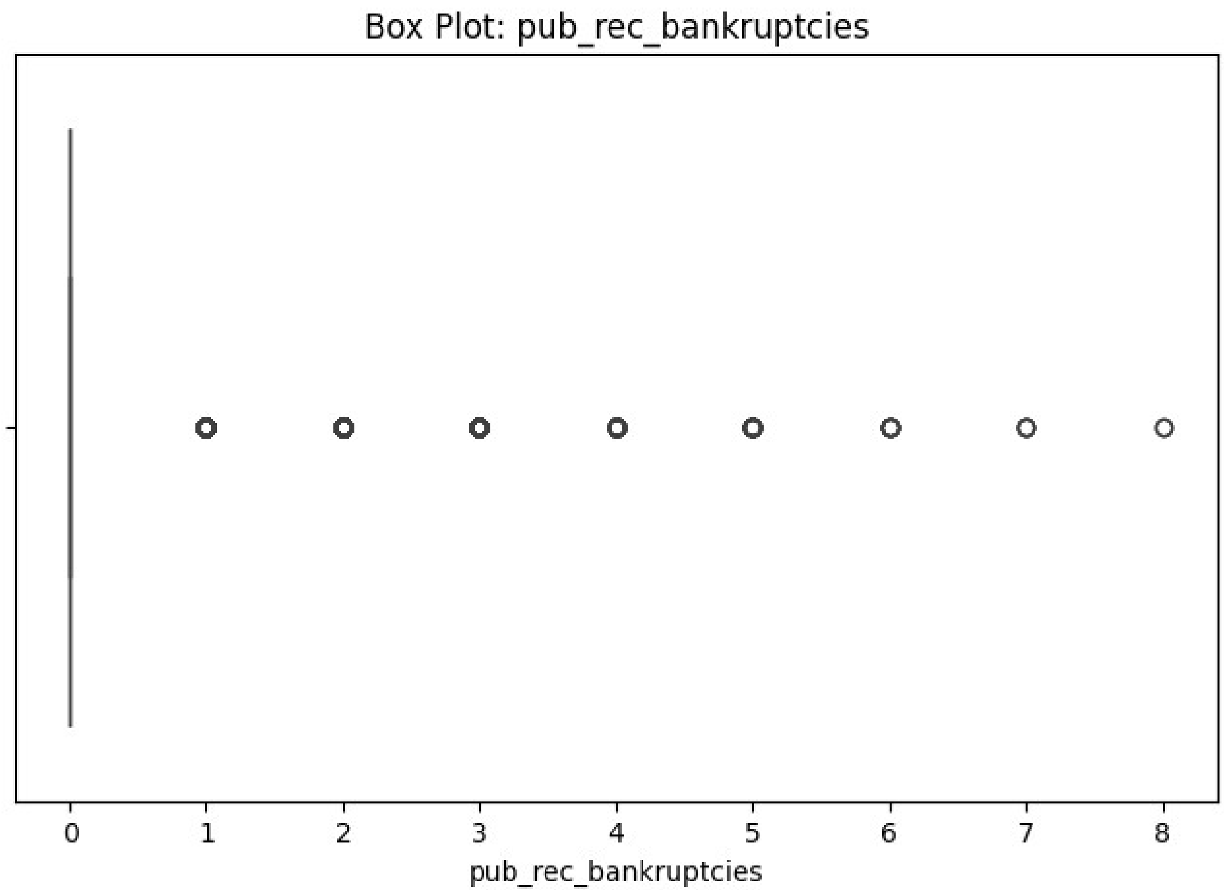

```
(6843, -4.5, 7.5)
```

These accounts are created when a borrower takes a mortgage loan from a lender, typically a bank or mortgage company.

Hence, This feature should not be treated

Bankruptcies 'pub_rec_bankruptcies'

```
# Extraction upper,lower bound & outlier
bankruptcies_outliers, bankruptcies_lower, bankruptcies_upper =
detect_outliers_iqr('pub_rec_bankruptcies')

plt.figure(figsize=(8, 5))
sns.boxplot(x=data['pub_rec_bankruptcies'])
plt.title('Box Plot: pub_rec_bankruptcies')
plt.show()

len(bankruptcies_outliers), bankruptcies_lower, bankruptcies_upper
```




```
(45115, 0.0, 0.0)
```

These are are legal filings that indicate an individual or business has declared bankruptcy and are publicly accessible documents.

Should not be treated

---

d. Encoding categorical columns

```
data[categorical_data].head(5)
```

```
         term grade sub_grade                emp_title emp_length  \
0   36 months     B        B4                Marketing  10+ years
1   36 months     B        B5           Credit analyst    4 years
2   36 months     B        B3             Statistician   < 1 year
3   36 months     A        A2          Client Advocate    6 years
4   60 months     C        C5  Destiny Management Inc.    9 years

  home_ownership verification_status   issue_d  loan_status  \
0           RENT        Not Verified  Jan-2015   Fully Paid
1       MORTGAGE        Not Verified  Jan-2015   Fully Paid
2           RENT     Source Verified  Jan-2015   Fully Paid
3           RENT        Not Verified  Nov-2014   Fully Paid
4       MORTGAGE            Verified  Apr-2013  Charged Off

              purpose                     title earliest_cr_line  \
0            vacation                  Vacation         Jun-1990
1  debt_consolidation        Debt consolidation         Jul-2004
2         credit_card  Credit card refinancing          Aug-2007
3         credit_card  Credit card refinancing          Sep-2006
4         credit_card    Credit Card Refinance          Mar-1999

  initial_list_status application_type  \
0                   w       INDIVIDUAL
1                   f       INDIVIDUAL
2                   f       INDIVIDUAL
3                   f       INDIVIDUAL
4                   f       INDIVIDUAL

                                              address
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
```

encoding 'term' column

```
print(data['term'].unique())
```

```
[' 36 months' ' 60 months']
```

```
from sklearn.preprocessing import OneHotEncoder

# Initialize OneHotEncoder
encoder = OneHotEncoder(sparse_output=False, drop='first')

# Transform the 'term' column
term_encoded = encoder.fit_transform(data[['term']])

# Create a DataFrame for the encoded data
data['term'] = pd.DataFrame(term_encoded, columns=encoder.get_feature_names_out(['term']))
```

Encoding 'grade'

```
data['grade'].value_counts().index
```

```
Index(['B', 'C', 'A', 'D', 'E', 'F', 'G'], dtype='object', name='grade')
```

```
# Define a mapping for grades
grade_mapping = {'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5, 'F': 6, 'G': 7}

# Apply the mapping to encode the grade column
data['grade'] = data['grade'].map(grade_mapping)
```

Encoding Sub_grade

```
data['sub_grade'].value_counts().sort_index().index
```

```
Index(['A1', 'A2', 'A3', 'A4', 'A5', 'B1', 'B2', 'B3', 'B4', 'B5', 'C1', 'C2',
       'C3', 'C4', 'C5', 'D1', 'D2', 'D3', 'D4', 'D5', 'E1', 'E2', 'E3', 'E4',
       'E5', 'F1', 'F2', 'F3', 'F4', 'F5', 'G1', 'G2', 'G3', 'G4', 'G5'],
      dtype='object', name='sub_grade')
```

```
# Define a mapping for grades
grade_mapping = {
        f"{grade}{num}": i + 1
        for i, (grade, num) in enumerate(
            [(g, n) for g in ['A', 'B', 'C', 'D', 'E', 'F', 'G'] for n in range(1, 6)]
        )
    }

# Apply the mapping to encode the grade column
data['sub_grade'] = data['sub_grade'].map(grade_mapping)
```

Encoding emp_title

```
print(data['emp_title'].value_counts())
```

```
emp_title
Unknown                      22927
Teacher                       4389
Manager                       4250
Registered Nurse              1856
RN                            1846
                             ...
```

```
Postman                            1
McCarthy & Holthus, LLC            1
jp flooring                        1
Histology Technologist             1
Gracon Services, Inc               1
Name: count, Length: 173106, dtype: int64
```

Performing Target encoding of this column

```
# loan_status is categorical, convert to numeric
data['loan_status'] = data['loan_status'].map({'Fully Paid': 1, 'Charged Off': 0})

data['loan_status']
```

```
0         1
1         1
2         1
3         1
4         0
         ..
396025    1
396026    1
396027    1
396028    1
396029    1
Name: loan_status, Length: 396030, dtype: int64
```

```
import category_encoders as ce

emp_title_target_mean = data.groupby('emp_title')['loan_status'].mean()

data['emp_title'] = data['emp_title'].map(emp_title_target_mean)

global_mean = data['loan_status'].mean()
data['emp_title']=data['emp_title'].fillna(global_mean)
```

```python
print(data[['emp_title', 'loan_status']])
```

```
        emp_title  loan_status
0        0.752809            1
1        0.666667            1
2        0.818182            1
3        1.000000            1
4        0.000000            0
...           ...          ...
396025   1.000000            1
396026   0.779570            1
396027   0.731343            1
396028   1.000000            1
396029   0.782609            1

[396030 rows x 2 columns]
```

---

emp_length

```python
data['emp_length'].value_counts().index
```

```
Index(['10+ years', '2 years', '< 1 year', '3 years', '5 years', '1 year',
       '4 years', '6 years', '7 years', '8 years', 'Not Provided', '9 years'],
      dtype='object', name='emp_length')
```

```python
emp_length_map={'10+ years':10, '2 years':2, '< 1 year': .5, '3 years':3, '5 years':5, '1 year':1,
       '4 years':4, '6 years':6, '7 years':7, '8 years':8, '9 years':9,'Not Provided':0.1}

data['emp_length'] = data['emp_length'].map(emp_length_map)
```

---

home_ownership encoding

```
data['home_ownership'].value_counts().index.sort_values()
```

```
Index(['ANY', 'MORTGAGE', 'NONE', 'OTHER', 'OWN', 'RENT'], dtype='object', name='home_ownership')
```

```
home_ownership_map={'ANY':1, 'MORTGAGE':13, 'NONE':14, 'OTHER':15, 'OWN':15.1, 'RENT':18}
data['home_ownership']=data['home_ownership'].map(home_ownership_map)
```

---

verification_status

```
data['verification_status'].value_counts().index.sort_values()
```

```
Index(['Not Verified', 'Source Verified', 'Verified'], dtype='object', name='verification_status')
```

```
verification_status_map={'Not Verified':0, 'Source Verified':1, 'Verified':0.5}
data['verification_status']=data['verification_status'].map(verification_status_map)
```

---

issue_d

```
data['issue_d'].value_counts().index.sort_values()
```

```
Index(['Apr-2008', 'Apr-2009', 'Apr-2010', 'Apr-2011', 'Apr-2012', 'Apr-2013',
       'Apr-2014', 'Apr-2015', 'Apr-2016', 'Aug-2007',
       ...
       'Sep-2007', 'Sep-2008', 'Sep-2009', 'Sep-2010', 'Sep-2011', 'Sep-2012',
       'Sep-2013', 'Sep-2014', 'Sep-2015', 'Sep-2016'],
      dtype='object', name='issue_d', length=115)
```

```
data['issue_d']=pd.to_datetime(data['issue_d'], format='%b-%Y')
```

```
data['issue_d'] =data['issue_d'].apply(lambda x: (x.year*10 + x.month)/1000)
```

---

purpose

```
data['purpose'].value_counts().index.sort_values()
```

```
Index(['car', 'credit_card', 'debt_consolidation', 'educational',
       'home_improvement', 'house', 'major_purchase', 'medical', 'moving',
       'other', 'renewable_energy', 'small_business', 'vacation', 'wedding'],
      dtype='object', name='purpose')
```

```
purpose_map={'car':1, 'credit_card':1.1, 'debt_consolidation':2, 'educational':3,
       'home_improvement':4, 'house':4.1, 'major_purchase':4.2, 'medical':4.3, 'moving':4.4,
       'other':5, 'renewable_energy':6, 'small_business':7, 'vacation':8, 'wedding':9}

data['purpose']=data['purpose'].map(purpose_map)
```

---

title encoding

```
data['title'].value_counts().index.sort_values()
```

```
Index(['\tcredit_card', '\tdebt_consolidation', '\tother', '\tsmall_business',
       '      debt consolidation', '   HITEK  EQUIPMENT ', '  A lending hand',
       '  Personal loan ', '  Three year debit free',
       '  debt consolidation cards and medical',
       ...
       'zero debt', 'zero dept', 'zero interest', 'zerodebt', 'zeusamoose',
       'zipcar', 'zonball Loan', 'zxcvb', '~Life Reorganization~',
       '~Summer Fun~'],
      dtype='object', name='title', length=48816)
```

performing Target Encoding

```
title_target_mean = data.groupby('title')['loan_status'].mean()

data['title'] = data['title'].map(title_target_mean)

global_mean = data['loan_status'].mean()
data['title']=data['title'].fillna(global_mean)
```

---

earliest_cr_line Encoding

```
data['earliest_cr_line'].value_counts().index.sort_values()
```

```
Index(['Apr-1955', 'Apr-1958', 'Apr-1960', 'Apr-1961', 'Apr-1962', 'Apr-1963',
       'Apr-1964', 'Apr-1965', 'Apr-1966', 'Apr-1967',
       ...
       'Sep-2004', 'Sep-2005', 'Sep-2006', 'Sep-2007', 'Sep-2008', 'Sep-2009',
       'Sep-2010', 'Sep-2011', 'Sep-2012', 'Sep-2013'],
      dtype='object', name='earliest_cr_line', length=684)
```

```
data['earliest_cr_line']=pd.to_datetime(data['earliest_cr_line'], format='%b-%Y')

data['earliest_cr_line'] =data['earliest_cr_line'].apply(lambda x: (x.year*10 + x.month)/1000)
```

---

initial_list_status Encoding

```
data['initial_list_status'].value_counts().index.sort_values()
```

```
Index(['f', 'w'], dtype='object', name='initial_list_status')
```

```
initial_list_status_map={'f':1,'w':0}
data['initial_list_status']=data['initial_list_status'].map(initial_list_status_map)
```

---

application_type Encoding

```
data['application_type'].value_counts().index.sort_values()
```

```
Index(['DIRECT_PAY', 'INDIVIDUAL', 'JOINT'], dtype='object', name='application_type')
```

```
application_type_map={'DIRECT_PAY':1, 'INDIVIDUAL':2, 'JOINT':3}
data['application_type']=data['application_type'].map(application_type_map)
```

---

address Encoding

```
data['address'].value_counts().index.sort_values()
```

```
Index(['000 Adam Station Apt. 329\r\nAshleyberg, AZ 22690',
       '000 Adrian Cliffs\r\nRandyton, LA 22690',
       '000 Alexandria Street\r\nPort Richard, FL 22690',
       '000 Amber Court\r\nLake Pamelatown, IN 00813',
       '000 Amy Pines Suite 498\r\nSouth Susan, ND 22690',
       '000 Anderson Hills Suite 654\r\nJensenchester, NH 29597',
       '000 Anderson Parks\r\nGrahamton, FL 30723',
       '000 Annette Fords\r\nKristenland, CA 11650',
       '000 April Island Suite 314\r\nLestad, IN 05113',
       '000 Barajas Place\r\nNew Kristenview, AR 30723',
       ...
       'Unit 9992 Box 2617\r\nDPO AA 05113',
       'Unit 9992 Box 7192\r\nDPO AA 22690',
       'Unit 9993 Box 6811\r\nDPO AP 30723',
```

```
       'Unit 9994 Box 8217\r\nDPO AP 30723',
       'Unit 9994 Box 9232\r\nDPO AP 48052',
       'Unit 9995 Box 6277\r\nDPO AE 48052',
       'Unit 9995 Box 8360\r\nDPO AP 00813',
       'Unit 9996 Box 9255\r\nDPO AP 05113',
       'Unit 9997 Box 3228\r\nDPO AA 11650',
       'Unit 9997 Box 3834\r\nDPO AP 86630'],
      dtype='object', name='address', length=393700)
```

using regex to extract state and zip code

```
import re

def encode_addres(data,col):
    data[col]=data[col].apply(lambda x: re.search(r'\b\d{5}\b',x).group() if re.search(r'\b\d{5}\b',x)
else "")
    return data[col]

data['address']=encode_addres(data,'address')
data['address']
```

```
0         22690
1         05113
2         87025
3         00813
4         11650
          ...
396025    12951
396026    05113
396027    70466
396028    29597
396029    48052
Name: address, Length: 396030, dtype: object
```

```
global_mean = data['loan_status'].mean()
data['address']=data['address'].fillna(global_mean)
```

All Categorical columns encoded

```
data[categorical_data].head(5)
```

```
   term  grade  sub_grade  emp_title  emp_length  home_ownership  \
0   0.0      2          9   0.752809        10.0            18.0
1   0.0      2         10   0.666667         4.0            13.0
2   0.0      2          8   0.818182         0.5            18.0
3   0.0      1          2   1.000000         6.0            18.0
4   1.0      3         15   0.000000         9.0            13.0

   verification_status  issue_d  loan_status  purpose     title  \
0                  0.0   20.151            1      8.0  0.794991
1                  0.0   20.151            1      2.0  0.770359
2                  1.0   20.151            1      1.1  0.807194
3                  0.0   20.151            1      1.1  0.807194
4                  0.5   20.134            0      1.1  0.910420

   earliest_cr_line  initial_list_status  application_type address
0            19.906                    0                 2   22690
1            20.047                    1                 2   05113
2            20.078                    1                 2   87025
3            20.069                    1                 2   00813
4            19.993                    1                 2   11650
```

Spliting data

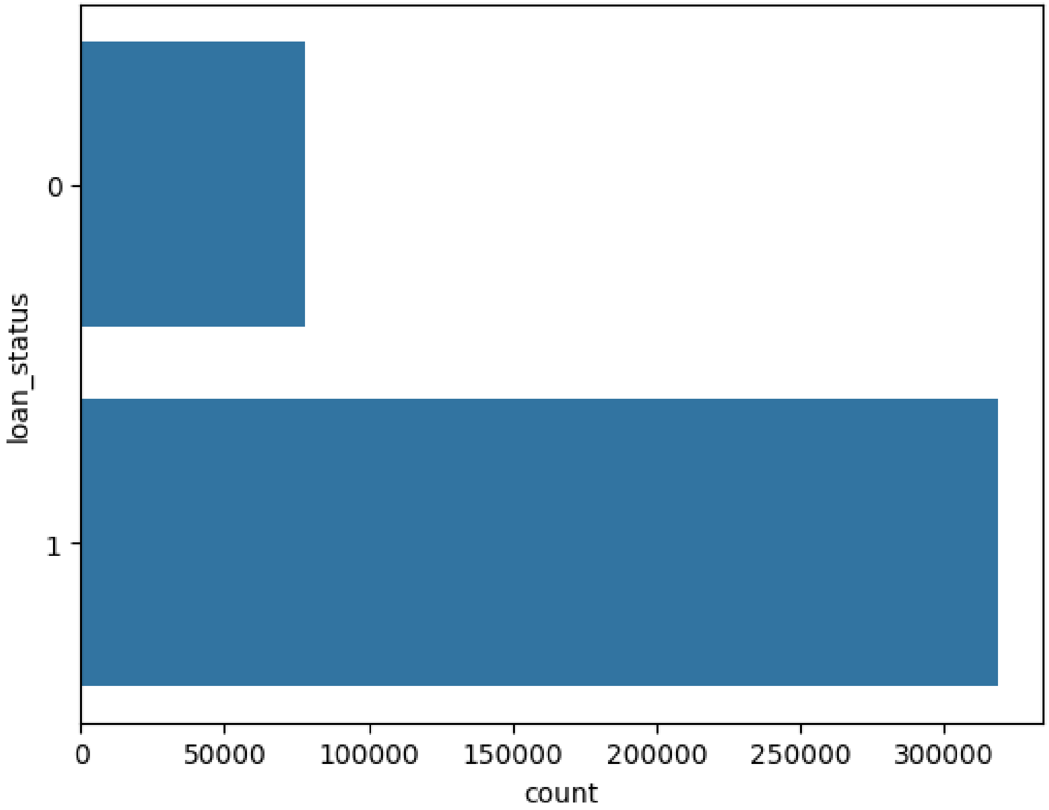
```
from sklearn.model_selection import train_test_split

y=data['loan_status']

X=data.drop('loan_status', axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, data['loan_status'], test_size=0.2,
random_state=0)
```

e. Check for imbalance dataset and balancing it

```
data['loan_status'].value_counts()
```

```
loan_status
1    318357
0     77673
Name: count, dtype: int64
```

```
sns.countplot(y=data['loan_status'])
plt.show()
```



as we can see the data is imbalaced

```
from imblearn.over_sampling import SMOTE

print('Before SMOTE:')
print(y_train.value_counts())
```

```
Before SMOTE:
loan_status
```

```
1    254546
0     62278
Name: count, dtype: int64

smt = SMOTE()

X_sm, y_sm = smt.fit_resample(X_train, y_train)

print('After Oversampling:')
print(y_sm.value_counts())

After Oversampling:
loan_status
0    254546
1    254546
Name: count, dtype: int64

data.to_csv('Bal_data.csv', index=False)
```

---

fitting balanced data on Logistic Regression Model

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score

model = LogisticRegression(C=5, penalty='l1', solver='liblinear')

model.fit(X_sm, y_sm)

LogisticRegression(C=5, penalty='l1', solver='liblinear')

train_pridiction_SMOTE=model.predict(X_sm)

test_prediction_SMOTE=model.predict(X_test)
```

```
print(f'Training F1 score: {round(f1_score(y_sm, train_pridiction_SMOTE)*100,2)}')
```

```
Training F1 score: 88.23
```

```
print(f'Test F1 score: {round(f1_score(y_test, test_prediction_SMOTE)*100,2)}')
```

```
Test F1 score: 91.39
```

f. Scaling

```
from sklearn.preprocessing import StandardScaler
```

```
data=pd.read_csv('Bal_data.csv')
```

```
data.head(10)
```

```
   loan_amnt  term  int_rate  installment  grade  sub_grade  emp_title  \
0    10000.0   0.0     11.44       329.48      2          9   0.752809
1     8000.0   0.0     11.99       265.68      2         10   0.666667
2    15600.0   0.0     10.49       506.97      2          8   0.818182
3     7200.0   0.0      6.49       220.65      1          2   1.000000
4    24375.0   1.0     17.27       609.33      3         15   0.000000
5    20000.0   0.0     13.33       677.07      3         13   0.793103
6    18000.0   0.0      5.32       542.07      1          1   1.000000
7    13000.0   0.0     11.14       426.47      2          7   0.812500
8    18900.0   1.0     10.99       410.84      2          8   0.857143
9    26300.0   0.0     16.29       928.40      3         15   1.000000

   emp_length  home_ownership  annual_inc  ...  open_acc  pub_rec  revol_bal  \
0        10.0            18.0    117000.0  ...      16.0      0.0    36369.0
1         4.0            13.0     65000.0  ...      17.0      0.0    20131.0
2         0.5            18.0     43057.0  ...      13.0      0.0    11987.0
3         6.0            18.0     54000.0  ...       6.0      0.0     5472.0
4         9.0            13.0     55000.0  ...      13.0      0.0    24584.0
5        10.0            13.0     86788.0  ...       8.0      0.0    25757.0
```

```
6         2.0            13.0    125000.0  ...       8.0       0.0     4178.0
7        10.0            18.0     46000.0  ...      11.0       0.0    13425.0
8        10.0            18.0    103000.0  ...      13.0       0.0    18637.0
9         3.0            13.0    115000.0  ...      13.0       0.0    22171.0

   revol_util  total_acc  initial_list_status  application_type  mort_acc  \
0        41.8       25.0                    0                 2       0.0
1        53.3       27.0                    1                 2       3.0
2        92.2       26.0                    1                 2       0.0
3        21.5       13.0                    1                 2       0.0
4        69.8       43.0                    1                 2       1.0
5       100.6       23.0                    1                 2       4.0
6         4.9       25.0                    1                 2       3.0
7        64.5       15.0                    1                 2       0.0
8        32.9       40.0                    0                 2       3.0
9        82.4       37.0                    1                 2       1.0

   pub_rec_bankruptcies  address
0                   0.0    22690
1                   0.0     5113
2                   0.0    87025
3                   0.0      813
4                   0.0    11650
5                   0.0    30723
6                   0.0    22690
7                   0.0    30723
8                   0.0    22690
9                   0.0      813

[10 rows x 27 columns]
```

```
scaler_std=StandardScaler()
```

```
scaled_data=data.drop('loan_status',axis=1)
```

```
scaled_data[scaled_data.columns]=scaler_std.fit_transform(scaled_data)

scaled_data['loan_status']=data['loan_status']
scaled_data
```

```
        loan_amnt      term  int_rate  installment     grade  sub_grade  \
0       -0.492295 -0.557975 -0.491799    -0.410815 -0.616534  -0.467127
1       -0.731683 -0.557975 -0.368816    -0.676342 -0.616534  -0.315634
2        0.177990 -0.557975 -0.704225     0.327874 -0.616534  -0.618620
3       -0.827438 -0.557975 -1.598649    -0.863751 -1.366267  -1.527580
4        1.228304  1.792196  0.811824     0.753882  0.133200   0.441833
...           ...       ...       ...          ...       ...        ...
396025  -0.492295  1.792196 -0.592422    -0.877360 -0.616534  -0.467127
396026   0.824337 -0.557975 -0.301734     1.132987  0.133200  -0.164140
396027  -1.090765 -0.557975 -0.816028    -1.110674 -0.616534  -0.921607
396028   0.824337  1.792196  0.373556     0.311435  0.133200  -0.012647
396029  -1.449846 -0.557975 -0.006574    -1.499142  0.133200  -0.012647

        emp_title  emp_length  home_ownership  annual_inc  ...   pub_rec  \
0       -0.194137    1.157532        1.176656    0.694330  ... -0.335785
1       -0.521648   -0.460950       -0.938189   -0.149311  ... -0.335785
2        0.054410   -1.405065        1.176656   -0.505312  ... -0.335785
3        0.745680    0.078544        1.176656   -0.327774  ... -0.335785
4       -3.056305    0.887785       -0.938189   -0.311550  ... -0.335785
...           ...         ...             ...         ...  ...       ...
396025   0.745680   -1.000444        1.176656   -0.554908  ... -0.335785
396026  -0.092392   -0.191203       -0.938189    0.580763  ... -0.335785
396027  -0.275749    1.157532        1.176656   -0.287214  ... -0.335785
396028   0.745680    1.157532       -0.938189   -0.165535  ... -0.335785
396029  -0.080839    1.157532        1.176656   -0.506301  ... -0.335785

        revol_bal  revol_util  total_acc  initial_list_status  \
0        0.996729   -0.491273  -0.034891            -1.227636
1        0.208163   -0.020083   0.133361             0.814574
2       -0.187334    1.573770   0.049235             0.814574
```

```
3       -0.503722   -1.323026  -1.044399             0.814574
4        0.424414    0.655973   1.479372             0.814574
...           ...         ...        ...                  ...
396025  -0.672818   -0.798571  -0.203142            -1.227636
396026   1.331523    1.717175  -1.465027             0.814574
396027   0.818746    0.537151  -0.203142             0.814574
396028  -0.006825    0.000404  -0.455519             0.814574
396029  -0.561026    1.536894  -0.539645             0.814574

        application_type  mort_acc  pub_rec_bankruptcies   address  \
0              -0.008284 -0.844172             -0.341282 -0.582082
1              -0.008284  0.614392             -0.341282 -1.218803
2              -0.008284 -0.844172             -0.341282  1.748430
3              -0.008284 -0.844172             -0.341282 -1.374569
4              -0.008284 -0.357984             -0.341282 -0.982003
...                  ...       ...                   ...       ...
396025         -0.008284 -0.844172             -0.341282 -0.934874
396026         -0.008284 -0.357984             -0.341282 -1.218803
396027         -0.008284 -0.844172             -0.341282  1.148586
396028         -0.008284  1.586769             -0.341282 -0.331879
396029         -0.008284 -0.357984             -0.341282  0.336647

        loan_status
0                 1
1                 1
2                 1
3                 1
4                 0
...             ...
396025            1
396026            1
396027            1
396028            1
396029            1
```

```
[396030 rows x 27 columns]
```

```
data.to_csv('scaled_data.csv', index=False)
```

3. Model building

a. Build the Logistic Regression model

```
Log_R_model=LogisticRegression(C=5, penalty='l1', solver='liblinear')
```

```
data=pd.read_csv('scaled_data.csv')
```

```
data.head(10)
```

```
   loan_amnt  term  int_rate  installment  grade  sub_grade  emp_title  \
0    10000.0   0.0     11.44       329.48      2          9   0.752809
1     8000.0   0.0     11.99       265.68      2         10   0.666667
2    15600.0   0.0     10.49       506.97      2          8   0.818182
3     7200.0   0.0      6.49       220.65      1          2   1.000000
4    24375.0   1.0     17.27       609.33      3         15   0.000000
5    20000.0   0.0     13.33       677.07      3         13   0.793103
6    18000.0   0.0      5.32       542.07      1          1   1.000000
7    13000.0   0.0     11.14       426.47      2          7   0.812500
8    18900.0   1.0     10.99       410.84      2          8   0.857143
9    26300.0   0.0     16.29       928.40      3         15   1.000000

   emp_length  home_ownership  annual_inc  ...  open_acc  pub_rec  revol_bal  \
0        10.0            18.0    117000.0  ...      16.0      0.0    36369.0
1         4.0            13.0     65000.0  ...      17.0      0.0    20131.0
2         0.5            18.0     43057.0  ...      13.0      0.0    11987.0
3         6.0            18.0     54000.0  ...       6.0      0.0     5472.0
4         9.0            13.0     55000.0  ...      13.0      0.0    24584.0
```

```
5        10.0            13.0     86788.0  ...       8.0       0.0    25757.0
6         2.0            13.0    125000.0  ...       8.0       0.0     4178.0
7        10.0            18.0     46000.0  ...      11.0       0.0    13425.0
8        10.0            18.0    103000.0  ...      13.0       0.0    18637.0
9         3.0            13.0    115000.0  ...      13.0       0.0    22171.0

   revol_util  total_acc  initial_list_status  application_type  mort_acc  \
0        41.8       25.0                    0                 2       0.0
1        53.3       27.0                    1                 2       3.0
2        92.2       26.0                    1                 2       0.0
3        21.5       13.0                    1                 2       0.0
4        69.8       43.0                    1                 2       1.0
5       100.6       23.0                    1                 2       4.0
6         4.9       25.0                    1                 2       3.0
7        64.5       15.0                    1                 2       0.0
8        32.9       40.0                    0                 2       3.0
9        82.4       37.0                    1                 2       1.0

   pub_rec_bankruptcies  address
0                   0.0    22690
1                   0.0     5113
2                   0.0    87025
3                   0.0      813
4                   0.0    11650
5                   0.0    30723
6                   0.0    22690
7                   0.0    30723
8                   0.0    22690
9                   0.0      813

[10 rows x 27 columns]
```

```
y=data['loan_status']
y
```

```
0         1
1         1
2         1
3         1
4         0
         ..
396025    1
396026    1
396027    1
396028    1
396029    1
Name: loan_status, Length: 396030, dtype: int64
```

```python
X=data.drop('loan_status',axis=1)
X.head(10)
```

```
   loan_amnt  term  int_rate  installment  grade  sub_grade  emp_title  \
0    10000.0   0.0     11.44       329.48      2          9   0.752809
1     8000.0   0.0     11.99       265.68      2         10   0.666667
2    15600.0   0.0     10.49       506.97      2          8   0.818182
3     7200.0   0.0      6.49       220.65      1          2   1.000000
4    24375.0   1.0     17.27       609.33      3         15   0.000000
5    20000.0   0.0     13.33       677.07      3         13   0.793103
6    18000.0   0.0      5.32       542.07      1          1   1.000000
7    13000.0   0.0     11.14       426.47      2          7   0.812500
8    18900.0   1.0     10.99       410.84      2          8   0.857143
9    26300.0   0.0     16.29       928.40      3         15   1.000000

   emp_length  home_ownership  annual_inc  ...  open_acc  pub_rec  revol_bal  \
0        10.0            18.0    117000.0  ...      16.0      0.0    36369.0
1         4.0            13.0     65000.0  ...      17.0      0.0    20131.0
2         0.5            18.0     43057.0  ...      13.0      0.0    11987.0
3         6.0            18.0     54000.0  ...       6.0      0.0     5472.0
4         9.0            13.0     55000.0  ...      13.0      0.0    24584.0
```

```
5        10.0            13.0     86788.0  ...       8.0      0.0    25757.0
6         2.0            13.0    125000.0  ...       8.0      0.0     4178.0
7        10.0            18.0     46000.0  ...      11.0      0.0    13425.0
8        10.0            18.0    103000.0  ...      13.0      0.0    18637.0
9         3.0            13.0    115000.0  ...      13.0      0.0    22171.0

   revol_util  total_acc  initial_list_status  application_type  mort_acc  \
0        41.8       25.0                    0                 2       0.0
1        53.3       27.0                    1                 2       3.0
2        92.2       26.0                    1                 2       0.0
3        21.5       13.0                    1                 2       0.0
4        69.8       43.0                    1                 2       1.0
5       100.6       23.0                    1                 2       4.0
6         4.9       25.0                    1                 2       3.0
7        64.5       15.0                    1                 2       0.0
8        32.9       40.0                    0                 2       3.0
9        82.4       37.0                    1                 2       1.0

   pub_rec_bankruptcies  address
0                   0.0    22690
1                   0.0     5113
2                   0.0    87025
3                   0.0      813
4                   0.0    11650
5                   0.0    30723
6                   0.0    22690
7                   0.0    30723
8                   0.0    22690
9                   0.0      813

[10 rows x 26 columns]
```

```
X.shape
```

```
(396030, 26)
```

```
X_train, X_test, y_train, y_test= train_test_split(X , y , test_size=0.2 , random_state=0)

X_train

        loan_amnt  term  int_rate  installment  grade  sub_grade  emp_title  \
44819     25000.0   1.0     14.83       592.52      4         18   0.740786
41622      9500.0   0.0     12.99       320.05      3         12   0.740786
362594     9000.0   0.0      8.39       283.65      2          6   0.813725
228739    16700.0   1.0     22.99       470.69      6         26   0.853301
210327     2800.0   0.0     15.80        98.17      3         13   1.000000
...           ...   ...       ...          ...    ...        ...        ...
359783    20000.0   0.0      7.89       625.72      1          5   0.768156
358083    12400.0   0.0     13.67       421.82      2         10   0.792481
152315    30000.0   0.0     12.69      1006.35      3         12   0.000000
117952    14000.0   0.0     14.31       480.60      3         14   0.740786
305711    12000.0   1.0     10.75       259.42      2          9   1.000000

        emp_length  home_ownership  annual_inc  ...  open_acc  pub_rec  \
44819         10.0            18.0    109000.0  ...      11.0      0.0
41622          0.1            13.0     40000.0  ...       6.0      1.0
362594        10.0            13.0     50000.0  ...      14.0      0.0
228739         7.0            13.0     68000.0  ...      13.0      0.0
210327         9.0            13.0    218554.0  ...      20.0      0.0
...            ...             ...         ...  ...       ...      ...
359783         1.0            13.0     70000.0  ...      11.0      1.0
358083        10.0            13.0     60000.0  ...       8.0      1.0
152315         1.0            13.0     80000.0  ...      16.0      1.0
117952         0.1            18.0     70000.0  ...       5.0      0.0
305711         2.0            18.0     80000.0  ...       6.0      0.0

        revol_bal  revol_util  total_acc  initial_list_status  \
44819      6390.0        41.5       39.0                    1
41622     62512.0        82.6       20.0                    1
362594     8835.0        23.2       20.0                    0
228739    23489.0        55.3       26.0                    1
```

```
210327    33014.0        90.7       44.0                     0
...           ...         ...        ...                   ...
359783     7888.0        21.2       26.0                     1
358083    16229.0        58.2       13.0                     0
152315    14300.0        66.2       28.0                     0
117952     6444.0        51.1       10.0                     0
305711    17573.0        43.7       17.0                     0

        application_type  mort_acc  pub_rec_bankruptcies  address
44819                  2       1.0                   0.0    93700
41622                  2       2.0                   1.0    48052
362594                 2       2.0                   0.0    16698
228739                 2       2.0                   0.0      813
210327                 2       7.0                   0.0      813
...                  ...       ...                   ...      ...
359783                 2       2.0                   1.0    70466
358083                 2       0.0                   1.0    39275
152315                 2       3.0                   0.0    48052
117952                 2       0.0                   0.0       38
305711                 2       1.0                   0.0    39726

[316824 rows x 26 columns]
```

```
y_train
```

```
44819     0
41622     0
362594    1
228739    1
210327    1
         ..
359783    1
358083    1
152315    0
117952    1
```

```
305711    1
Name: loan_status, Length: 316824, dtype: int64
```

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

```
LogisticRegression(C=5, penalty='l1', solver='liblinear')
```

```
train_prediction=Log_R_model.predict(X_train)
```

```
test_prediction=Log_R_model.predict(X_test)
```

```
print(f'Training F1 score: {round(f1_score(y_sm, train_pridiction_SMOTE)*100, 2)}')
```

```
Training F1 score: 88.23
```

```
print(f'Testing F1 score: {round(f1_score(y_test, test_prediction)*100, 2)}')
```

```
Testing F1 score: 93.94
```

---

b. Display model coefficients with column names

```
coef_data=pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': Log_R_model.coef_.flatten()
})
```

```
coef_data
```

```
                 Feature   Coefficient
0              loan_amnt -5.839863e-06
1                   term -4.420362e-01
2               int_rate  4.814400e-02
3            installment -2.039812e-04
4                  grade -6.988649e-02
```

```
5               sub_grade -8.326825e-02
6               emp_title  8.681530e+00
7              emp_length  3.256475e-02
8          home_ownership -5.794522e-02
9              annual_inc  9.554786e-08
10    verification_status -1.562678e-01
11                issue_d -1.026119e-01
12                purpose  3.819968e-02
13                  title  7.455429e+00
14                    dti -2.239198e-02
15       earliest_cr_line -9.517767e-02
16               open_acc -2.823216e-02
17                pub_rec -7.767415e-02
18              revol_bal  1.812769e-06
19             revol_util -6.305601e-03
20              total_acc  4.394986e-03
21    initial_list_status -6.586571e-02
22       application_type -6.762682e-01
23               mort_acc  3.017343e-02
24   pub_rec_bankruptcies  1.012863e-01
25                address -2.071532e-05
```

---

4. Results Evaluation

a. Confusion Matrix and comments

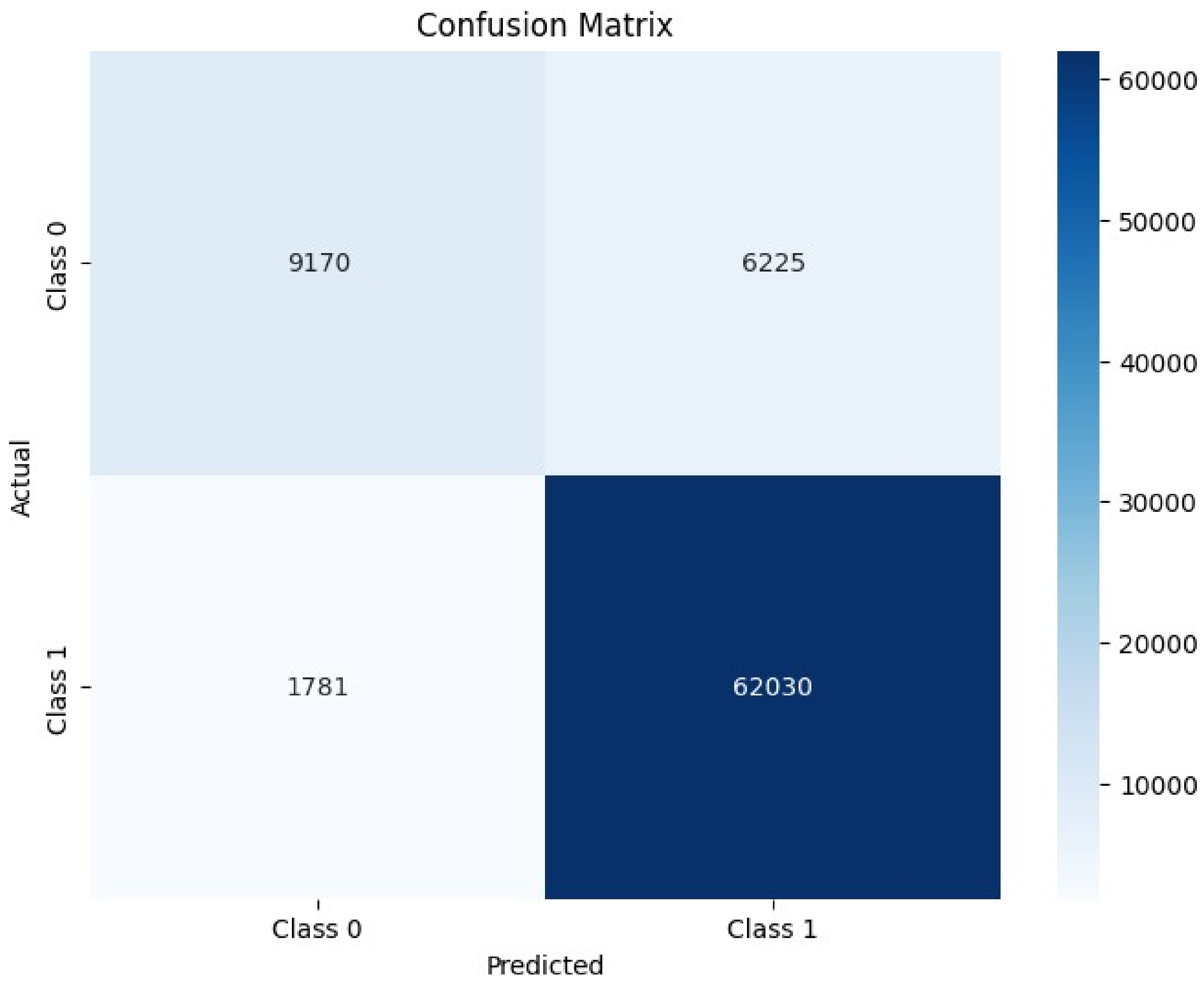
```python
from sklearn.metrics import confusion_matrix

conf_matrix= confusion_matrix(y_test, test_prediction)
```

```
conf_matrix
```

```
array([[ 9170,  6225],
       [ 1781, 62030]], dtype=int64)
```

```
TN=conf_matrix[0,0]
FP=conf_matrix[0,1]
TP=conf_matrix[1,1]
FN=conf_matrix[1,0]
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Class 0', 'Class 1'],
yticklabels=['Class 0', 'Class 1'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



True Negetives : - 9159

- The Model correctly predicted the 0 class

False Positives : - 6236

- Model incorrectly predicted class 1 for actual class 0.

True Positives : -62051

- Model correctly predicted class 1 (positive cases).

False Negatives : - 1,760

- Model failed to predict class 1 for actual class 1.

Performance Metrics Derived

```
1 > Accuracy

accuracy = (TP + TN) / (TP + TN + FP + FN)
print(f'Proportion of correct predictions : {round(accuracy*100,2)}')

Proportion of correct predictions : 89.89

2 > Precision

precision = TP / (TP + FP)
print(f'Proportion of positive predictions that are correct : {round(precision*100,2)}')

Proportion of positive predictions that are correct : 90.88

3 > Recall

recall = TP / (TP + FN)
print(f'Proportion of actual positives correctly identified : {round(recall*100,2)}')

Proportion of actual positives correctly identified : 97.21

4 > F1 Score
```

```
f1_score = 2 * (precision * recall) / (precision + recall)
print(f'Harmonic mean of precision and recall : {round(recall*100,2)}')

Harmonic mean of precision and recall : 97.21
```

High Recall (97.22%):

- The model performs well in identifying actual positive cases.

Good Precision (90.87%):

- Most positive predictions are correct.

---

b. Classification Report and comments

```
from sklearn.metrics import classification_report

report = classification_report(y_test, test_prediction, target_names=['Fully Paid', 'Charged Off'])
print(report)

              precision    recall  f1-score   support

  Fully Paid       0.84      0.60      0.70     15395
 Charged Off       0.91      0.97      0.94     63811

    accuracy                           0.90     79206
   macro avg       0.87      0.78      0.82     79206
weighted avg       0.89      0.90      0.89     79206


        Overall Accuracy: 90%
                        Indicates that 90% of the model's predictions are correct.
```

```
        Class-Specific Metrics:
```

Fully Paid:

```
    Precision: 0.84 → Out of all cases predicted as "Fully Paid," 84% are correct.
    Recall: 0.59 → The model identifies only 59% of the actual "Fully Paid" cases.
    F1-Score: 0.70 → A relatively low score due to the imbalance between precision and recall.
```

Charged Off:

```
    Precision: 0.91 → The model is very confident in predicting "Charged Off" cases.
    Recall: 0.97 → The model identifies almost all "Charged Off" cases.
    F1-Score: 0.94 → A strong balance between precision and recall for this class.
```

Macro Avg:

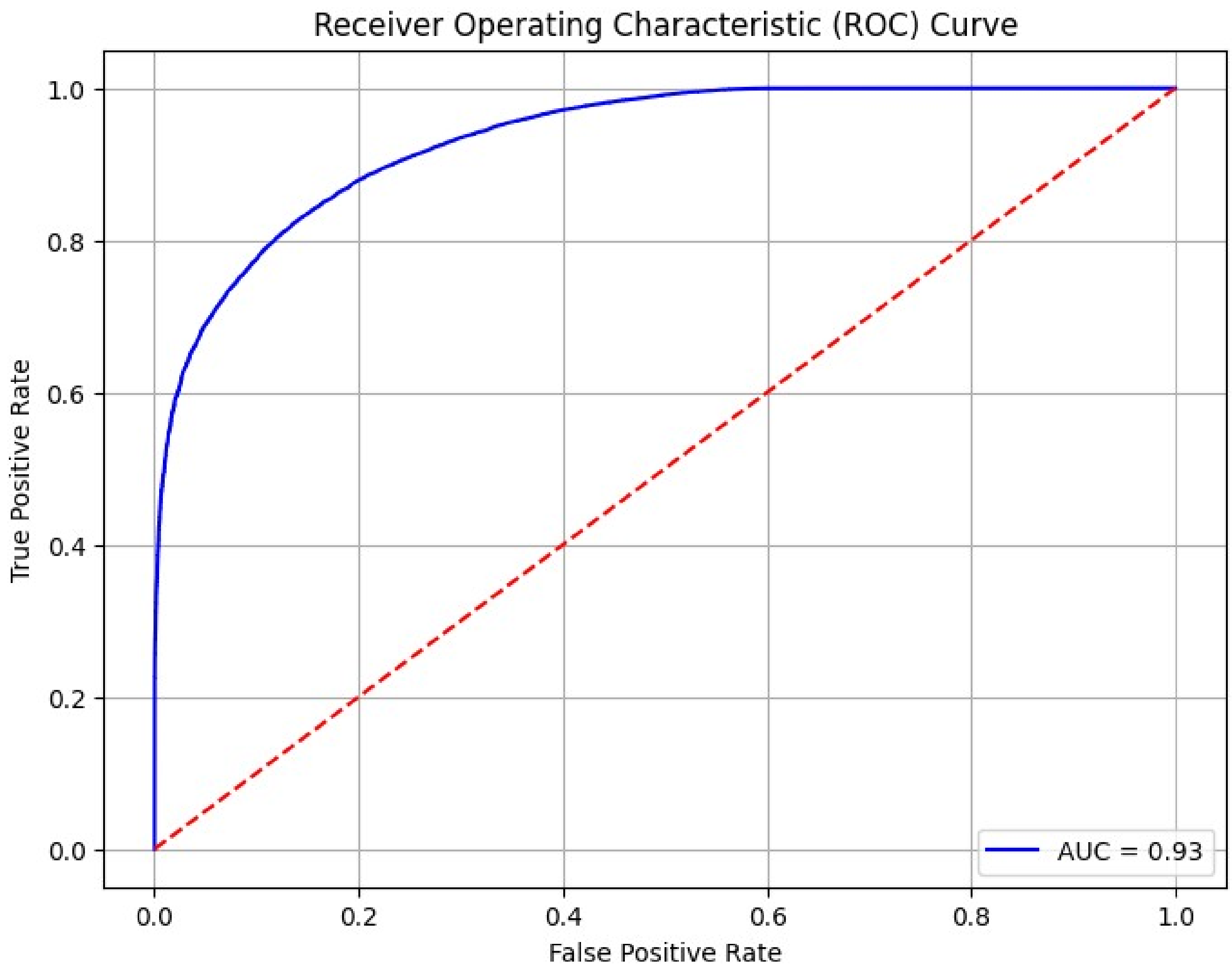
```
    Precision (0.87), Recall (0.78), and F1-Score (0.82) reflect the unweighted average across both
classes.
    The lower recall (0.78) indicates that the model struggles with the minority class.
```

Weighted Avg:

```
    These averages are weighted by the support (i.e., the number of samples in each class).
    Indicates overall model performance:
    Precision: 0.90
    Recall: 0.90
    F1-Score: 0.89

            Low Recall for "Fully Paid" (0.59):

    - The model fails to identify 41% of the actual "Fully Paid" cases, leading to many false negatives.
    - This may be problematic if "Fully Paid" is a critical class for the analysis.
```

c. AU-ROC Curve & comments

```
from sklearn.metrics import roc_curve, auc

y_pred_proba = Log_R_model.predict_proba(X_test)[:, 1]

fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.grid()
plt.show()
```



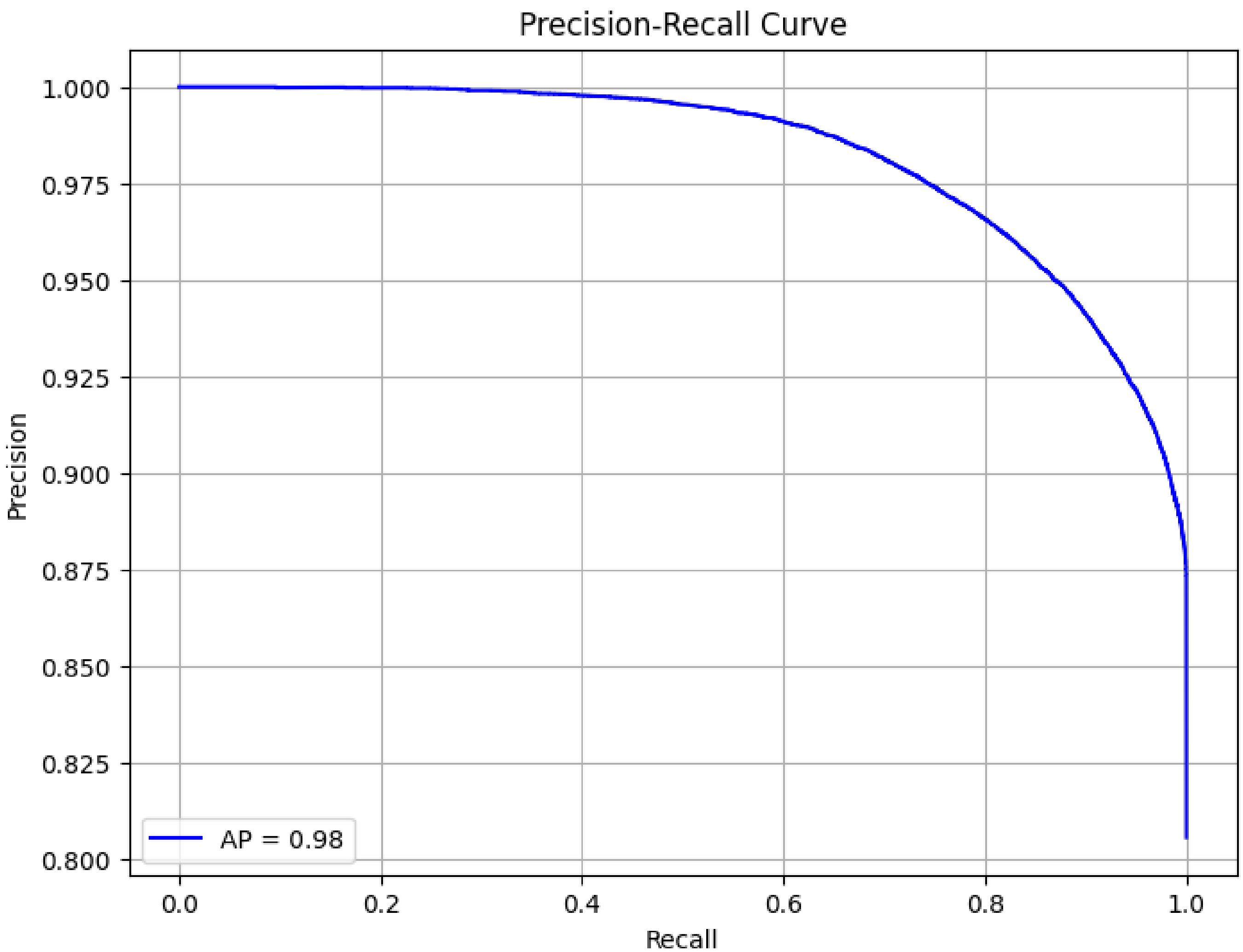

```
Indicates a strong model.
```

```
The classifier has good separation between positive and negative classes.

AUC = 0.93 <br>
    Excellent classifier; it can distinguish between the two classes effectively.
```

---

d. Precision Recall Curve & comments

```python
from sklearn.metrics import precision_recall_curve, average_precision_score

precision, recall, thresholds = precision_recall_curve(y_test, y_pred_proba)

average_precision = average_precision_score(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(recall, precision, color='blue', label=f'AP = {average_precision:.2f}')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend(loc='lower left')
plt.grid()
plt.show()
```




```
Has a precision-recall curve that reaches the top-right corner (precision = 1, recall = 1)
```

```
 AP = 0.98 is excellent. <br>
    Indicates that the model maintains a good balance between precision and recall across different
thresholds.
```
