Nama : Davit Cany Agho

Nim : G.211.21.0116

1. Importing Libraries

```
import os
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

2. Importing and Exploration of the dataset

```
df = pd.read_csv('loans.csv', index_col = 'client_id')
df.head()
```

client_id	loan_type	loan_amount	repaid	loan_id	loan_start	loan_end	rate
46109	home	13672	0	10243	2002-04-16	2003-12-20	2.15
46109	credit	9794	0	10984	2003-10-21	2005-07-17	1.25
46109	home	12734	1	10990	2006-02-01	2007-07-05	0.68
46109	cash	12518	1	10596	2010-12-08	2013-05-05	1.24
46109	credit	14049	1	11415	2010-07-07	2012-05-21	3.13

```
df.shape
```

```
(443, 7)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 443 entries, 46109 to 26945
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   loan_type    443 non-null    object
 1   loan_amount  443 non-null    int64
 2   repaid       443 non-null    int64
 3   loan_id      443 non-null    int64
 4   loan_start   443 non-null    object
 5   loan_end     443 non-null    object
 6   rate         443 non-null    float64
dtypes: float64(1), int64(3), object(3)
memory usage: 27.7+ KB
```

3. Cheking the datatypes of the columns

```
df.dtypes
```

```
loan_type       object
loan_amount      int64
repaid           int64
loan_id          int64
loan_start      object
loan_end        object
rate           float64
dtype: object
```

4. Converting the data types columns

```
df['loan_id'] = df['loan_id'].astype('object')
df['repaid'] = df['repaid'].astype('category')
df['loan_start'] = pd.to_datetime(df['loan_start'], format = '%Y-%m-%d')
df['loan_end'] = pd.to_datetime(df['loan_end'], format = '%Y-%m-%d')
```

```
df.dtypes
```

```
loan_type              object
loan_amount             int64
```

```
repaid              category
loan_id               object
loan_start    datetime64[ns]
loan_end      datetime64[ns]
rate                 float64
dtype: object
```

5. Summary Statistic of the data

```
1 df.describe()
```

	loan_amount	rate
count	443.000000	443.000000
mean	7982.311512	3.217156
std	4172.891992	2.397168
min	559.000000	0.010000
25%	4232.500000	1.220000
50%	8320.000000	2.780000
75%	11739.000000	4.750000
max	14971.000000	12.620000

```
1 df.describe(exclude=[np.number])
```

	loan_type	repaid	loan_id	loan_start	loan_end
count	443	443.0	443.0	443	443
unique	4	2.0	443.0	430	428
top	home	1.0	10243.0	2007-05-16 00:00:00	2008-08-29 00:00:00
freq	121	237.0	1.0	2	2
first	NaN	NaN	NaN	2000-01-26 00:00:00	2001-08-02 00:00:00
last	NaN	NaN	NaN	2014-11-11 00:00:00	2017-05-07 00:00:00

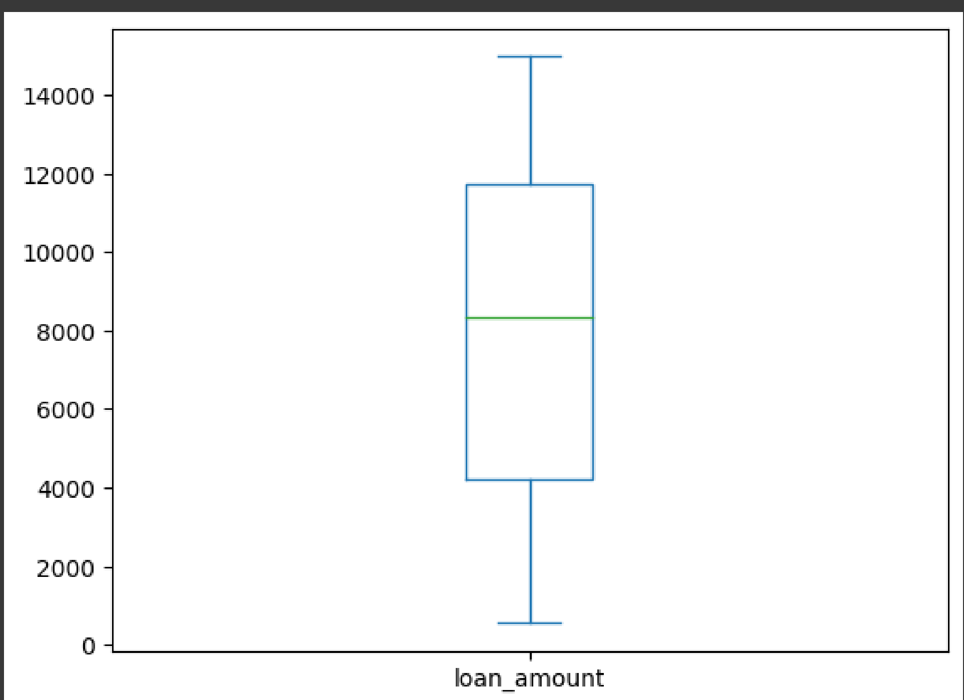
6. Missing values

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

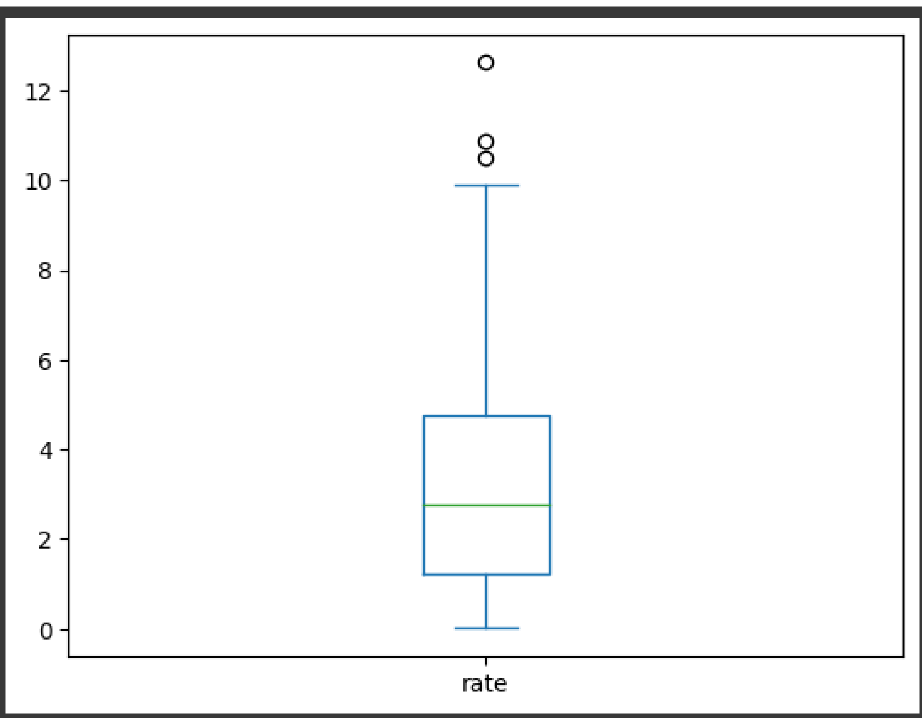
```
loan_type      0
loan_amount    0
repaid         0
loan_id        0
loan_start     0
loan_end       0
rate           0
dtype: int64
```

7. Outliers Treatment

```
1 df['loan_amount'].plot(kind='box')
2 plt.show()
```




```
1 df['rate'].plot(kind='box')
2 plt.show()
```



8. Transformation

8a. SQRT transformation

```
df['SQRT_RATE'] = df['rate']**0.5
```

```
df['sqrt_rate'] = np.sqrt(df['rate'])
```

```
df.head()
```

client_id	loan_type	loan_amount	repaid	loan_id	loan_start	loan_end	rate	SQRT_RATE	sqrt_rate
46109	home	13672	0	10243	2002-04-16	2003-12-20	2.15	1.466288	1.466288
46109	credit	9794	0	10984	2003-10-21	2005-07-17	1.25	1.118034	1.118034
46109	home	12734	1	10990	2006-02-01	2007-07-05	0.68	0.824621	0.824621
46109	cash	12518	1	10596	2010-12-08	2013-05-05	1.24	1.113553	1.113553
46109	credit	14049	1	11415	2010-07-07	2012-05-21	3.13	1.769181	1.769181

```
print("The skewness of the original data is {}".format(df.rate.skew()))
print("The skewness of the SQRT transformed data is {}".format(df.SQRT_RATE.skew()))

print('')

print("The kurtosis of the original data is {}".format(df.rate.kurt()))
print("The kurtosis of the SQRT transformed data is {}".format(df.SQRT_RATE.kurt()))
```

```
The skewness of the original data is 0.884204614329943
The skewness of the SQRT transformed data is 0.04964154055528862

The kurtosis of the original data is 0.42437165143736433
The kurtosis of the SQRT transformed data is -0.6318437642052039
```

```
fig, axes = plt.subplots(1,2, figsize=(15,5))
sns.distplot(df['rate'], ax=axes[0])
sns.distplot(df['sqrt_rate'], ax=axes[1])

plt.show()
```

8b. Log Transformation

```
df['Log Rate'] = np.log(df['rate'])
```

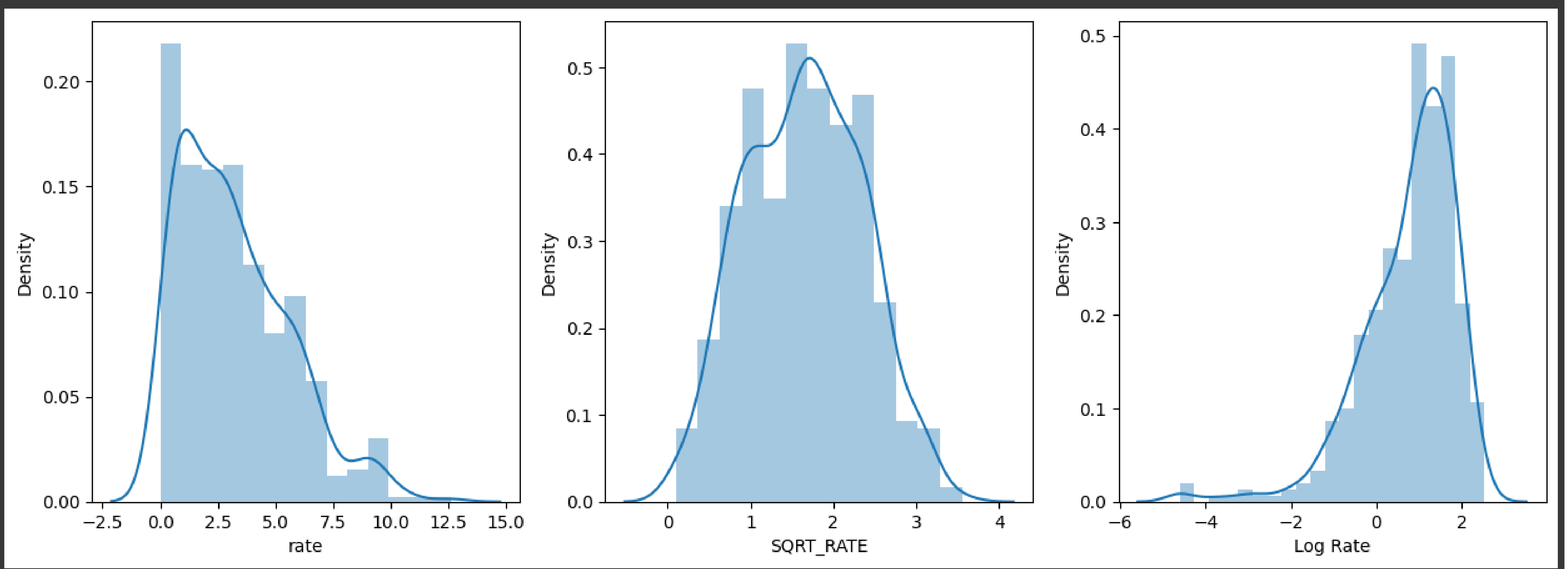
```
df.head()
```

client_id	loan_type	loan_amount	repaid	loan_id	loan_start	loan_end	rate	SQRT_RATE	sqrt_rate	Log Rate
46109	home	13672	0	10243	2002-04-16	2003-12-20	2.15	1.466288	1.466288	0.765468
46109	credit	9794	0	10984	2003-10-21	2005-07-17	1.25	1.118034	1.118034	0.223144
46109	home	12734	1	10990	2006-02-01	2007-07-05	0.68	0.824621	0.824621	-0.385662
46109	cash	12518	1	10596	2010-12-08	2013-05-05	1.24	1.113553	1.113553	0.215111
46109	credit	14049	1	11415	2010-07-07	2012-05-21	3.13	1.769181	1.769181	1.141033

```
print("The skewness of the original data is {}".format(df.rate.skew()))
print("The skewness of the SQRT transformed data is {}".format(df.SQRT_RATE.skew()))
print("The skewness of the LOG transformed data is {}".format(df['Log Rate'].skew()))

print('')

print("The kurtosis of the original data is {}".format(df.rate.kurt()))
print("The kurtosis of the SQRT transformed data is {}".format(df.SQRT_RATE.kurt()))
print("The kurtosis of the LOG transformed data is {}".format(df['Log Rate'].kurt()))
```

```
The skewness of the original data is 0.884204614329943
The skewness of the SQRT transformed data is 0.04964154055528862
The skewness of the LOG transformed data is -1.5943217626331552

The kurtosis of the original data is 0.42437165143736433
The kurtosis of the SQRT transformed data is -0.6318437642052039
The kurtosis of the LOG transformed data is 4.157026150198228
```

```
fig, axes = plt.subplots(1,3,figsize=(15,5))

sns.distplot(df['rate'], ax=axes[0])
sns.distplot(df['SQRT_RATE'], ax=axes[1])
sns.distplot(df['Log Rate'], ax=axes[2])

plt.show()
```




```
df['LOG_Rate'] = df['rate'].apply(lambda x:np.log(x))
```

```
df.head()
```

client_id	loan_type	loan_amount	repaid	loan_id	loan_start	loan_end	rate	SQRT_RATE	sqrt_rate	Log Rate	LOG_Rate
46109	home	13672	0	10243	2002-04-16	2003-12-20	2.15	1.466288	1.466288	0.765468	0.765468
46109	credit	9794	0	10984	2003-10-21	2005-07-17	1.25	1.118034	1.118034	0.223144	0.223144
46109	home	12734	1	10990	2006-02-01	2007-07-05	0.68	0.824621	0.824621	-0.385662	-0.385662
46109	cash	12518	1	10596	2010-12-08	2013-05-05	1.24	1.113553	1.113553	0.215111	0.215111
46109	credit	14049	1	11415	2010-07-07	2012-05-21	3.13	1.769181	1.769181	1.141033	1.141033

Outliers Treatment using Capping Approach

1) Z-Score approach to treat Outliers:

```
df1 = pd.read_csv('loans.csv', index_col = 'client_id')
df1.head()
```

client_id	loan_type	loan_amount	repaid	loan_id	loan_start	loan_end	rate
46109	home	13672	0	10243	2002-04-16	2003-12-20	2.15
46109	credit	9794	0	10984	2003-10-21	2005-07-17	1.25
46109	home	12734	1	10990	2006-02-01	2007-07-05	0.68
46109	cash	12518	1	10596	2010-12-08	2013-05-05	1.24
46109	credit	14049	1	11415	2010-07-07	2012-05-21	3.13

```
df1['loan_id'] = df1['loan_id'].astype('object')
df1['repaid'] = df1['repaid'].astype('category')
```

```
df1['loan_start'] = pd.to_datetime(df1['loan_start'], format = '%Y-%m-%d')
df1['loan_end'] = pd.to_datetime(df1['loan_end'], format = '%Y-%m-%d')
```

```
import scipy.stats as stats
```

Using SciPy Library to calculate the Z-Score:

```
df1['ZR'] = stats.zscore(df1['rate'])
```

```
df1.head()
```

client_id	loan_type	loan_amount	repaid	loan_id	loan_start	loan_end	rate	ZR
46109	home	13672	0	10243	2002-04-16	2003-12-20	2.15	-0.445677
46109	credit	9794	0	10984	2003-10-21	2005-07-17	1.25	-0.821544
46109	home	12734	1	10990	2006-02-01	2007-07-05	0.68	-1.059594
46109	cash	12518	1	10596	2010-12-08	2013-05-05	1.24	-0.825721
46109	credit	14049	1	11415	2010-07-07	2012-05-21	3.13	-0.036399

```
df1[(df1['ZR']<-3) | (df1['ZR']>3)]
```

client_id	loan_type	loan_amount	repaid	loan_id	loan_start	loan_end	rate	ZR
41480	credit	2947	1	10302	2005-11-10	2008-03-16	10.49	3.037362
48177	other	6318	0	10224	2003-02-02	2005-05-08	10.89	3.204415
49624	home	8133	1	10312	2009-03-14	2011-03-21	12.62	3.926916

```
df1[(df1['ZR']<-3) | (df1['ZR']>3)].shape[0]
```

3

```
df2 = df1[(df1['ZR']>-3) & (df1['ZR']<3)].reset_index()
df2.head()
```

	client_id	loan_type	loan_amount	repaid	loan_id	loan_start	loan_end	rate	ZR
0	46109	home	13672	0	10243	2002-04-16	2003-12-20	2.15	-0.445677
1	46109	credit	9794	0	10984	2003-10-21	2005-07-17	1.25	-0.821544
2	46109	home	12734	1	10990	2006-02-01	2007-07-05	0.68	-1.059594
3	46109	cash	12518	1	10596	2010-12-08	2013-05-05	1.24	-0.825721
4	46109	credit	14049	1	11415	2010-07-07	2012-05-21	3.13	-0.036399

```
df1.shape
```

(443, 8)

```
df2.shape
```

(440, 9)

```
df3 = df2.copy()
```

```
df3.drop(columns = ['ZR'], inplace=True)
df3.head()
```

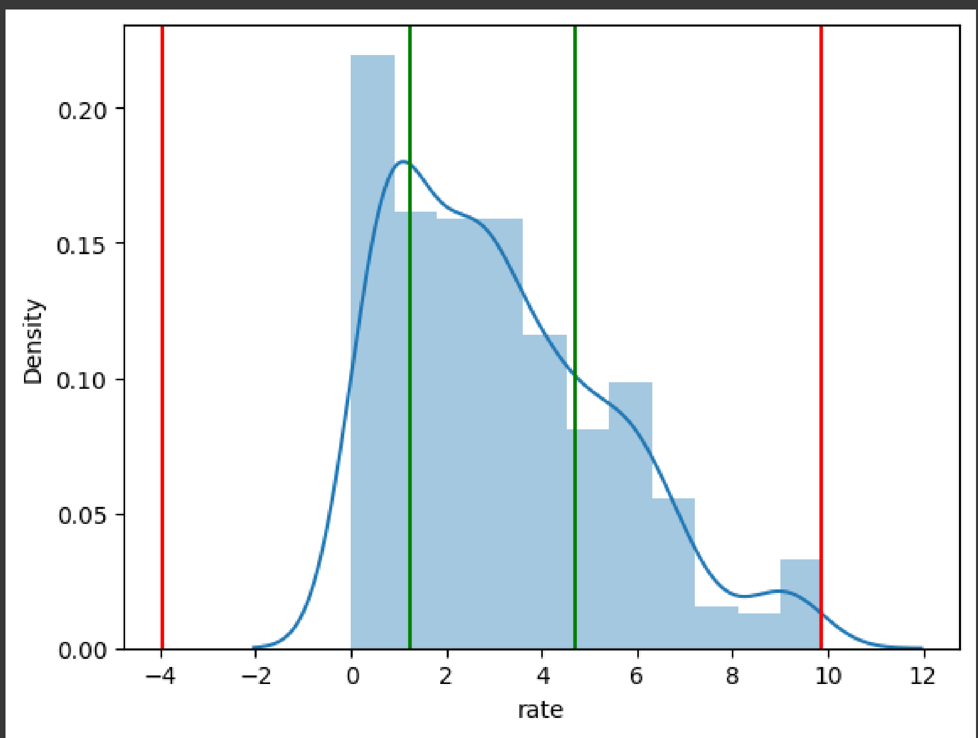
	client_id	loan_type	loan_amount	repaid	loan_id	loan_start	loan_end	rate
0	46109	home	13672	0	10243	2002-04-16	2003-12-20	2.15
1	46109	credit	9794	0	10984	2003-10-21	2005-07-17	1.25
2	46109	home	12734	1	10990	2006-02-01	2007-07-05	0.68
3	46109	cash	12518	1	10596	2010-12-08	2013-05-05	1.24
4	46109	credit	14049	1	11415	2010-07-07	2012-05-21	3.13

2) IQR Method to treat Outliers

```
Q1 = df3.rate.quantile(0.25)
Q2 = df3.rate.quantile(0.50)
Q3 = df3.rate.quantile(0.75)

IQR = Q3 - Q1

LC = Q1 - (1.5*IQR)

UC = Q3 + (1.5*IQR)

display(LC)
display(UC)
```

```
-3.9762499999999994
9.87375
```

```
sns.distplot(df3.rate)
plt.axvline(UC, color='r')
plt.axvline(LC, color='r')
plt.axvline(Q1, color='g')
plt.axvline(Q3, color='g')
plt.show()
```




```
df3[(df3.rate<LC) | (df3.rate>UC)].reset_index(drop=True)
```

	client_id	loan_type	loan_amount	repaid	loan_id	loan_start	loan_end	rate
0	39505	cash	11647	1	11928	2003-07-28	2005-12-24	9.91

```
df3[(df3.rate<LC) | (df3.rate>UC)].shape[0]
```

```
1
```

```
df4 = df3[(df3.rate>LC) & (df3.rate<UC)]
df4.head()
```

```
df3.shape
```

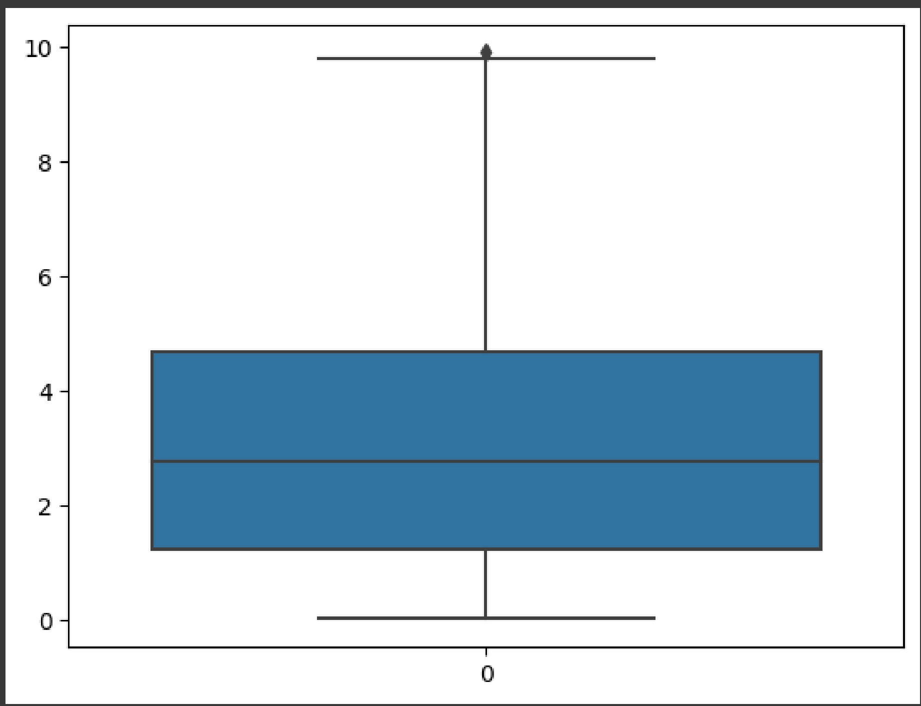
```
(440, 8)
```

```
df4.shape
```

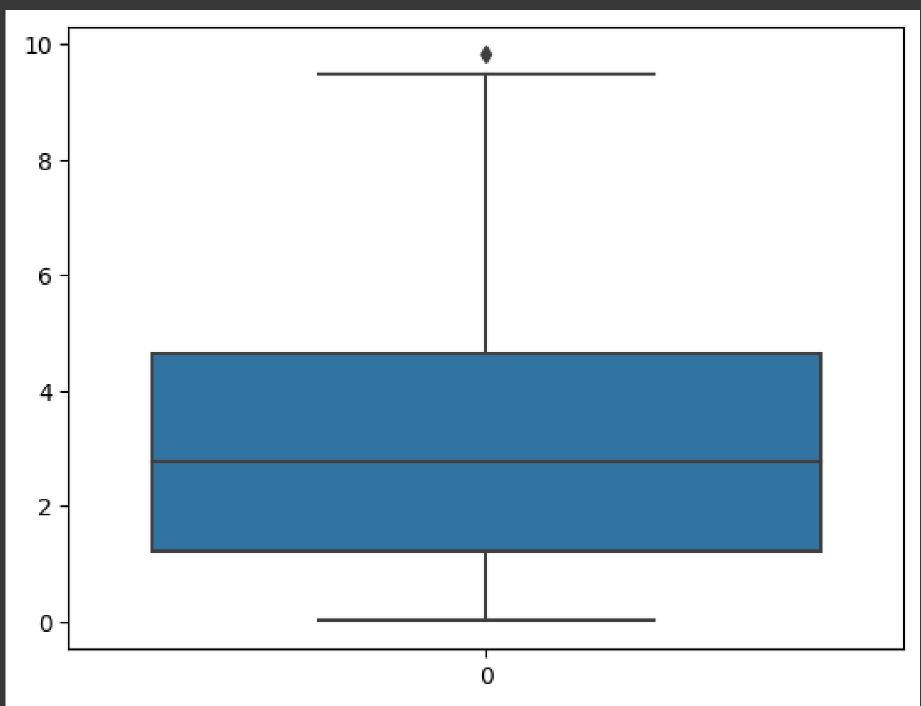
```
(439, 8)
```

4 46109 credit 14049 1 11415 2010-07-07 2012-05-21 3.13

```
sns.boxplot(df2.rate)
plt.show()
```




```
sns.boxplot(df4.rate)
plt.show()
```



9. Scaling the Numerical Features

9a. Standardization (Z-Score)

```
avg_rate = df3['rate'].mean()
avg_rate
```

```
3.161818181818182
```

```
std_rate = df3['rate'].std()
std_rate
```

```
2.3079474188229154
```

```
df3['Z_Score_Rate'] = (df3['rate'] - avg_rate)/std_rate
```

```
df3.head()
```

	client_id	loan_type	loan_amount	repaid	loan_id	loan_start	loan_end	rate	Z_Score_Rate
0	46109	home	13672	0	10243	2002-04-16	2003-12-20	2.15	-0.438406

```
print("The skewness of the original data is {}".format(df3.rate.skew()))
print("The kurtois for the original data is {}".format(df3.rate.kurt()))

print('')

print("The skewness for the Zscore Scaled column is {}".format(df3.Z_Score_Rate.skew()))
print("The kurtois for the Zscore Scaled Column is {}".format(df3.Z_Score_Rate.kurt()))
```

```
The skewness of the original data is 0.7594062707815686
The kurtois for the original data is -0.05964248048746912

The skewness for the Zscore Scaled column is 0.7594062707815691
The kurtois for the Zscore Scaled Column is -0.05964248048746823
```

```
avg_LA = df3['loan_amount'].mean()
avg_LA
```

```
7997.195454545455
```

```
std_LA = df3['loan_amount'].std()
std_LA
```

```
4179.435966237437
```

```
df3['Z_Score_LA'] = (df3['loan_amount'] - avg_LA)/std_LA
```

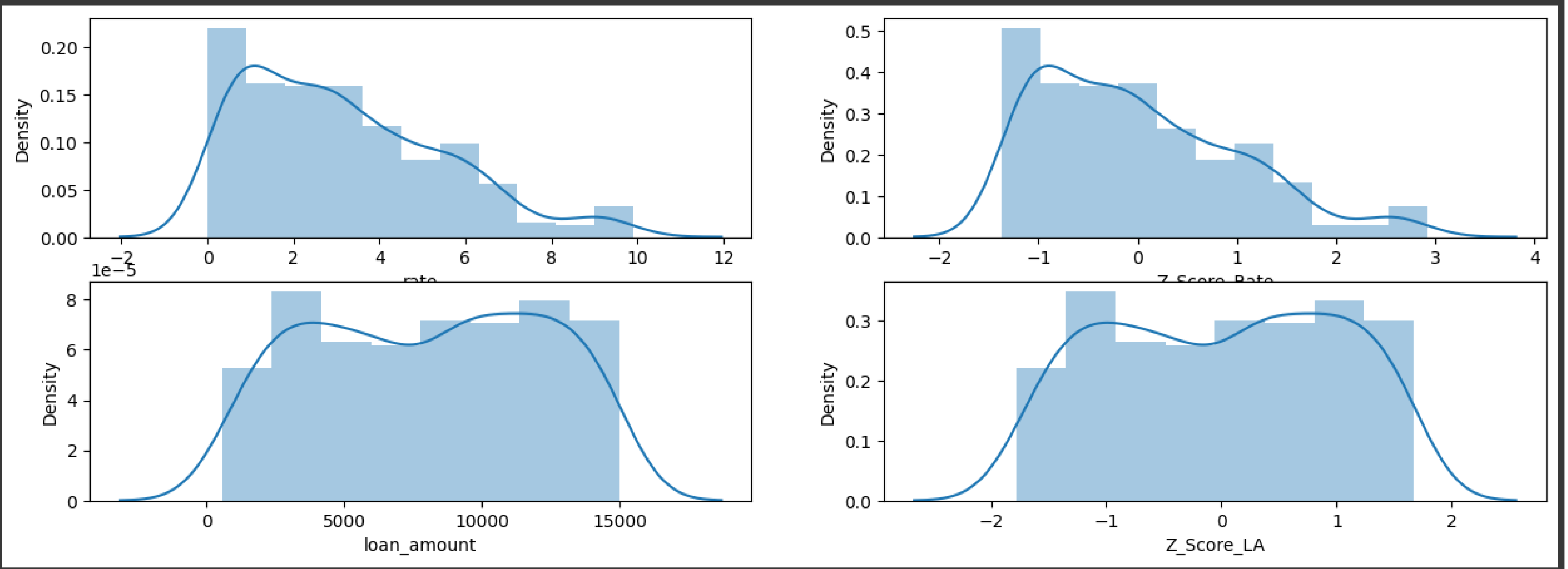
```
df3.head()
```

	client_id	loan_type	loan_amount	repaid	loan_id	loan_start	loan_end	rate	Z_Score_Rate	Z_Score_LA
0	46109	home	13672	0	10243	2002-04-16	2003-12-20	2.15	-0.438406	1.357792
1	46109	credit	9794	0	10984	2003-10-21	2005-07-17	1.25	-0.828363	0.429916
2	46109	home	12734	1	10990	2006-02-01	2007-07-05	0.68	-1.075336	1.133360
3	46109	cash	12518	1	10596	2010-12-08	2013-05-05	1.24	-0.832696	1.081678
4	46109	credit	14049	1	11415	2010-07-07	2012-05-21	3.13	-0.013786	1.447996

```
print("The skewness of the original data is {}".format(df3.loan_amount.skew()))
print("The kurtois for the original data is {}".format(df3.loan_amount.kurt()))

print('')

print("The skewness for the Zscore Scaled column is {}".format(df3.Z_Score_LA.skew()))
print("The kurtois for the Zscore Scaled Column is {}".format(df3.Z_Score_LA.kurt()))
```

```
The skewness of the original data is -0.04678765472024289
The kurtois for the original data is -1.2354309429278456

The skewness for the Zscore Scaled column is -0.04678765472024289
The kurtois for the Zscore Scaled Column is -1.2354309429278456
```

```
fig, axes = plt.subplots(2,2, figsize=(15,5))

sns.distplot(df3['rate'], ax=axes[0,0])
sns.distplot(df3['Z_Score_Rate'], ax=axes[0,1])
sns.distplot(df3['loan_amount'], ax=axes[1,0])
sns.distplot(df3['Z_Score_LA'], ax=axes[1,1])

plt.show()
```




```
df4 = df3.copy()
```

```
df4 = df3.copy()
df4.drop(columns = ['Z_Score_Rate'], inplace=True)
df4.head()
```

	client_id	loan_type	loan_amount	repaid	loan_id	loan_start	loan_end	rate	Z_Score_LA
0	46109	home	13672	0	10243	2002-04-16	2003-12-20	2.15	1.357792
1	46109	credit	9794	0	10984	2003-10-21	2005-07-17	1.25	0.429916
2	46109	home	12734	1	10990	2006-02-01	2007-07-05	0.68	1.133360
3	46109	cash	12518	1	10596	2010-12-08	2013-05-05	1.24	1.081678
4	46109	credit	14049	1	11415	2010-07-07	2012-05-21	3.13	1.447996

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

```
df4_num = df[['loan_amount','rate']]
df4_num.head()
```

client_id	loan_amount	rate
46109	13672	2.15
46109	9794	1.25
46109	12734	0.68
46109	12518	1.24
46109	14049	3.13

```
SS = StandardScaler()

scaled_x = SS.fit_transform(df4_num)
scaled_x
```

```
       [ 9.97243153e-01,  6.40162675e-01],
       [ 5.74516639e-01, -1.03035976e+00],
       [ 2.60710577e-01,  1.07867481e+00],
       [-1.25841785e+00,  1.23319814e+00],
       [ 7.81368667e-02, -1.20994092e+00],
       [ 3.97940751e-01, -5.58437173e-01],
       [-1.34166762e+00, -7.81619725e-02],
       [ 1.10184517e+00,  6.56867900e-01],
       [ 1.24867186e+00, -5.25026724e-01],
       [-1.46421599e-01,  6.94454654e-01],
       [ 1.10448422e+00, -4.03913848e-01],
       [ 1.66971899e+00, -5.33379336e-01],
       [-1.67850532e+00, -1.09300435e+00],
       [-6.92703253e-01,  7.44570327e-01],
       [ 1.47471692e-01,  1.73017856e+00],
       [-1.61037006e+00,  2.75754986e+00],
       [ 1.10088552e+00, -9.59362558e-01],
       [-5.33641006e-01, -5.58437173e-01],
       [-4.87770528e-02,  2.50697150e+00],
       [ 1.36143092e+00,  1.02855914e+00],
       [ 7.77243032e-01,  9.99324999e-01],
       [ 1.22348101e+00,  3.10234494e-01],
       [ 5.13338851e-01, -9.13423191e-01],
       [-3.16999665e-01, -8.75836436e-01],
       [ 9.09435035e-01,  1.24155075e+00],
       [-1.44291086e+00, -1.05124129e+00],
       [-9.36694663e-01, -1.33940641e+00],
       [-5.82583236e-01,  5.77518084e-01],
       [ 1.60662190e+00,  1.36683993e+00],
       [-1.29800348e+00, -3.91384929e-01],
       [ 1.53229602e-01, -1.55173811e-02],
       [ 1.25682890e+00,  3.01881882e-01],
       [ 7.76523293e-01,  2.30884678e-01],
       [ 9.50220226e-01,  1.26243228e+00],
       [-1.34862510e+00,  4.27171065e-01],
       [ 3.15170803e-01,  9.30665773e-02],
       [ 3.61521107e-02,  3.92691557e+00],
       [-4.63586442e-01,  9.72428834e-02],
       [ 1.20740685e+00, -1.20994092e+00],
       [-2.55055023e-02,  1.09120373e+00],
       [-1.32463381e+00,  3.77055391e-01],
       [ 1.54472437e+00, -1.33940641e+00],
       [ 9.03917038e-01, -1.08047543e+00],
       [-4.73182957e-01, -2.41037910e-01],
       [ 8.50943405e-02, -8.29897069e-01],
       [ 3.03894897e-01, -1.49159176e-01],
       [ 2.19925385e-01,  3.18587106e-01],
       [-1.58709851e+00,  1.22300720e-01],
       [ 1.68584027e-01, -1.90922237e-01],
       [-1.07296519e+00, -1.13410750e-02],
       [-1.75839632e+00, -1.11572421e-01],
       [ 1.37798491e+00, -8.42425987e-01],
       [ 1.58598939e+00, -1.13476741e+00],
       [ 1.19493138e+00, -3.16211420e-01],
       [-1.50048996e+00,  8.57330592e-01],
       [ 3.23087929e-01,  1.01603022e+00],
       [-9.08145029e-01,  5.35755023e-01],
       [-1.04105677e+00, -1.28929074e+00]])
```

6b. Normalization: Min Max Scalar

```
min_rate = df4.rate.min()
min_rate
```

```
0.01
```

```
max_rate = df4.rate.max()
max_rate
```

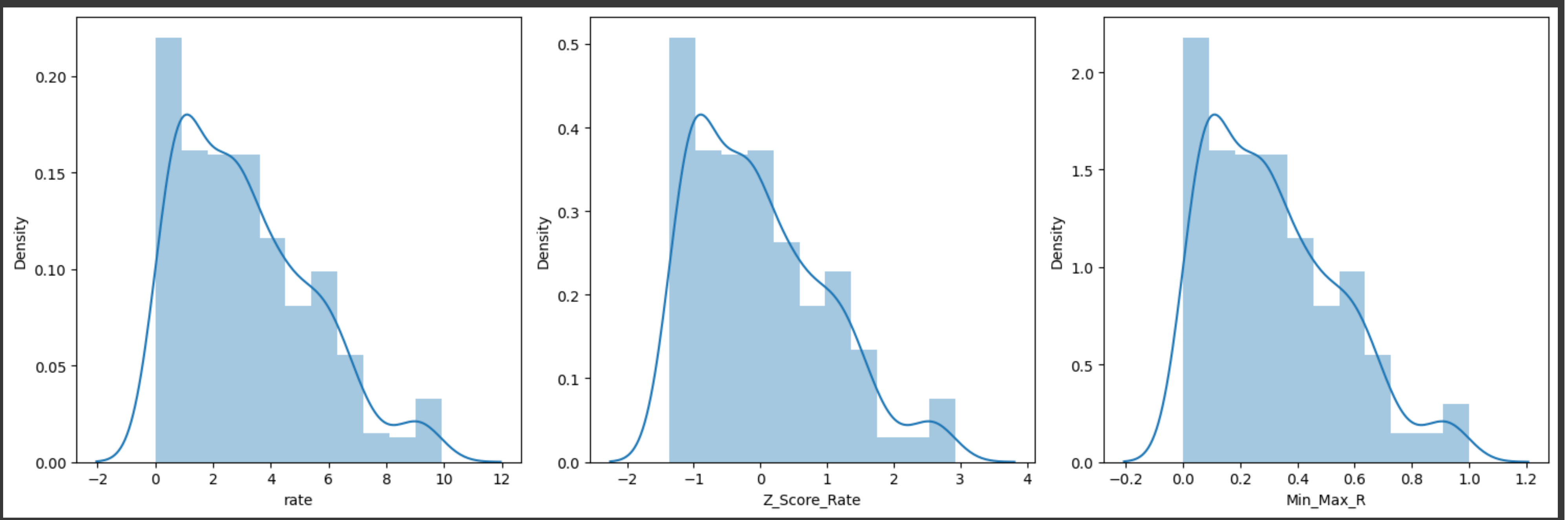
```
9.91
```

```
df4['Min_Max_R'] = (df4['rate'] - min_rate)/ (max_rate - min_rate)
```

```
print("The skewness for the original data is {}.".format(df4.rate.skew()))
print("The skewness for the Zscore Scaled column is {}.".format(df3.Z_Score_Rate.skew()))
print("The skewness for the Min Max Scaled Data is {}.".format(df4.Min_Max_R.skew()))

print('')

print("The kurtosis for the original data is {}.".format(df4.rate.kurt()))
print("The kurtosis for the Zscore Scaled column is {}.".format(df3.Z_Score_Rate.kurt()))
print("The kurtosis for the Min Max Scaled Data is {}.".format(df4.Min_Max_R.kurt()))

```

```
The skewness for the original data is 0.7594062707815686.
The skewness for the Zscore Scaled column is 0.7594062707815691.
The skewness for the Min Max Scaled Data is 0.7594062707815686.

The kurtosis for the original data is -0.05964248048746912.
The kurtosis for the Zscore Scaled column is -0.05964248048746823.
The kurtosis for the Min Max Scaled Data is -0.05964248048746823.
```

```
fig, axes = plt.subplots(1,3, figsize=(15,5))

sns.distplot(df3['rate'],ax=axes[0])
sns.distplot(df3['Z_Score_Rate'],ax=axes[1])
sns.distplot(df4['Min_Max_R'],ax=axes[2])

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




```
min_LA = df4.loan_amount.min()
min_LA
```

```
559
```

```
max_LA = df4.loan_amount.max()
max_LA
```

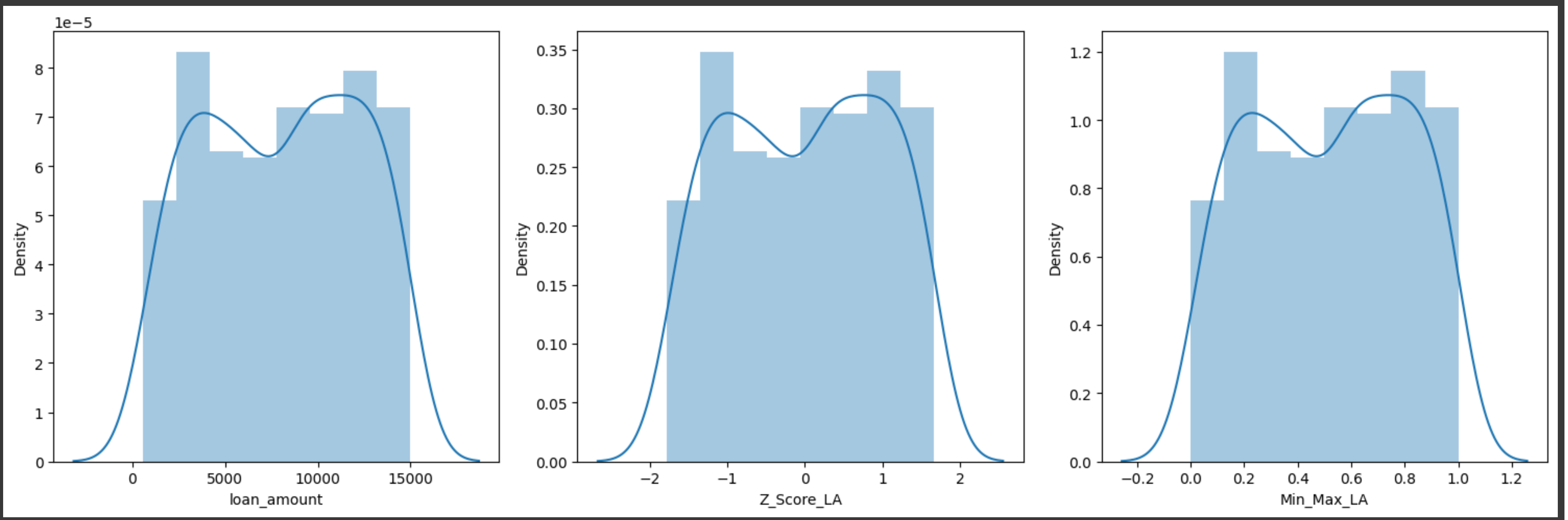
```
14971
```

```
df4['Min_Max_LA'] = (df4['loan_amount'] - min_LA)/ (max_LA - min_LA)
```

```
print("The skewness for the original data is {}.".format(df4.loan_amount.skew()))
print("The skewness for the Zscore Scaled column is {}.".format(df3.Z_Score_LA.skew()))
print("The skewness for the Min Max Scaled Data is {}.".format(df4.Min_Max_LA.skew()))

print('')

print("The kurtosis for the original data is {}.".format(df4.loan_amount.kurt()))
print("The kurtosis for the Zscore Scaled column is {}.".format(df3.Z_Score_LA.kurt()))
print("The kurtosis for the Min Max Scaled Data is {}.".format(df4.Min_Max_LA.kurt()))

```

```
The skewness for the original data is -0.04678765472024289.
The skewness for the Zscore Scaled column is -0.04678765472024289.
The skewness for the Min Max Scaled Data is -0.04678765472024256.

The kurtosis for the original data is -1.2354309429278456.
The kurtosis for the Zscore Scaled column is -1.2354309429278456.
The kurtosis for the Min Max Scaled Data is -1.2354309429278452.
```

```
fig, axes = plt.subplots(1,3, figsize=(15,5))

sns.distplot(df3['loan_amount'],ax=axes[0])
sns.distplot(df3['Z_Score_LA'],ax=axes[1])
sns.distplot(df4['Min_Max_LA'],ax=axes[2])

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




```
from sklearn.preprocessing import MinMaxScaler
```

```
MS = MinMaxScaler()

MinMaxScaled = MS.fit_transform(df4_num)
MinMaxScaled
```

```
       [6.52234249e-03, 2.33148295e-01],
       [9.13613655e-01, 9.43695480e-02],
       [9.73771857e-01, 3.88580492e-02],
       [8.60671663e-01, 1.94290246e-01],
       [8.11129614e-02, 4.17129262e-01],
       [6.08520677e-01, 4.47264076e-01],
       [2.52428532e-01, 3.56066614e-01],
       [2.13988343e-01, 9.51625694e-03]])
```

10. Encoding the Categorical Features

```
df_loans = df3.copy()
```

```
df_loans.drop(columns = ['Z_Score_Rate'], inplace=True)
df_loans.drop(columns = ['Z_Score_LA'], inplace=True)
```

```
df_loans.head()
```

	client_id	loan_type	loan_amount	repaid	loan_id	loan_start	loan_end	rate
0	46109	home	13672	0	10243	2002-04-16	2003-12-20	2.15
1	46109	credit	9794	0	10984	2003-10-21	2005-07-17	1.25
2	46109	home	12734	1	10990	2006-02-01	2007-07-05	0.68
3	46109	cash	12518	1	10596	2010-12-08	2013-05-05	1.24
4	46109	credit	14049	1	11415	2010-07-07	2012-05-21	3.13

```
df_loans.dtypes
```

```
client_id                int64
loan_type               object
loan_amount              int64
repaid                category
loan_id                 object
loan_start     datetime64[ns]
loan_end       datetime64[ns]
rate                   float64
dtype: object
```

```
df_loans.repaid.head()
```

```
0    0
1    0
2    1
3    1
4    1
Name: repaid, dtype: category
Categories (2, int64): [0, 1]
```

1) pd.get_dummies approach:

```
dummy_cat = pd.get_dummies(df_loans['loan_type'], drop_first = True)
dummy_cat.head()
```

	credit	home	other
0	0	1	0
1	1	0	0
2	0	1	0
3	0	0	0
4	1	0	0

2. OneHot Encoding

```
from sklearn.preprocessing import OneHotEncoder
```

```
EO_tips = OneHotEncoder(drop = 'first').fit(df_loans[['loan_type']])
EO_tips.categories_
```

```
[array(['cash', 'credit', 'home', 'other'], dtype=object)]
```

3. Label Encoding

```
from sklearn.preprocessing import LabelEncoder
```

```
LE = LabelEncoder()
LE_tips = LE.fit(df_loans[['loan_type']])
```

```
LE_tips.classes_
```

```
array(['cash', 'credit', 'home', 'other'], dtype=object)
```

```
LE_tips.transform(['other', 'cash', 'home', 'credit'])
```

```
array([3, 0, 2, 1])
```

```
LE_tips.inverse_transform([1,2,3,0])
```

```
array(['credit', 'home', 'other', 'cash'], dtype=object)
```

11. Creating new Derived Features

```
import datetime as dt
```

```
df_loans['loan_tenure'] = df_loans['loan_end'] - df_loans['loan_start']
```

```
df_loans.head()
```

	client_id	loan_type	loan_amount	repaid	loan_id	loan_start	loan_end	rate	loan_tenure
0	46109	home	13672	0	10243	2002-04-16	2003-12-20	2.15	613 days
1	46109	credit	9794	0	10984	2003-10-21	2005-07-17	1.25	635 days
2	46109	home	12734	1	10990	2006-02-01	2007-07-05	0.68	519 days
3	46109	cash	12518	1	10596	2010-12-08	2013-05-05	1.24	879 days
4	46109	credit	14049	1	11415	2010-07-07	2012-05-21	3.13	684 days

```
df_loans.dtypes
```

```
client_id                int64
loan_type               object
loan_amount              int64
repaid                category
loan_id                 object
loan_start      datetime64[ns]
loan_end        datetime64[ns]
rate                   float64
loan_tenure    timedelta64[ns]
dtype: object
```

```
df_loans['loan_tenure'] = df_loans['loan_tenure']/365
df_loans['loan_tenure']
```

```
0     1 days 16:18:24.657534246
1     1 days 17:45:12.328767123
2     1 days 10:07:33.698630136
3     2 days 09:47:50.136986301
4     1 days 20:58:31.232876712
                 ...
435   2 days 13:01:09.041095890
436             1 days 09:36:00
437   2 days 14:20:03.287671232
438   1 days 17:37:18.904109589
439   1 days 17:57:02.465753424
Name: loan_tenure, Length: 440, dtype: timedelta64[ns]
```

12. Training and Testing data

```
from sklearn.model_selection import train_test_split
```

```
Y = df_loans['loan_amount']
X = df_loans.drop('loan_amount', axis=1)
```

```
X.head()
```

	client_id	loan_type	repaid	loan_id	loan_start	loan_end	rate	loan_tenure
0	46109	home	0	10243	2002-04-16	2003-12-20	2.15	1 days 16:18:24.657534246
1	46109	credit	0	10984	2003-10-21	2005-07-17	1.25	1 days 17:45:12.328767123
2	46109	home	1	10990	2006-02-01	2007-07-05	0.68	1 days 10:07:33.698630136
3	46109	cash	1	10596	2010-12-08	2013-05-05	1.24	2 days 09:47:50.136986301
4	46109	credit	1	11415	2010-07-07	2012-05-21	3.13	1 days 20:58:31.232876712

```
Y.head()
```

```
0    13672
1     9794
```

```
2    12734
3    12518
4    14049
Name: loan_amount, dtype: int64
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,train_size=0.8, random_state=0)

print("The shape of X_train is:", X_train.shape)
print("The shape of X_test is:", X_test.shape)

print('')
print("The shape of Y_train is:", Y_train.shape)
print("The shape of Y_test is:", Y_test.shape)

```

```
The shape of X_train is: (352, 8)
The shape of X_test is: (88, 8)

The shape of Y_train is: (352,)
The shape of Y_test is: (88,)
```

```
median_y_train = Y_train.median()
median_y_test = Y_test.median()
```

```
print('The median for Y Train variables is:',median_y_train)
```

```
The median for Y Train variables is: 8412.5
```

```
print('The median for Y test variables is:',median_y_test)
```

```
The median for Y test variables is: 7673.0
```