In [1]:
```
# depencies
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

pd.set_option("display.max_columns",None)
```

In [2]:
```
# loading the data
df = pd.read_csv(r"loan_data.csv")
df.head()
```

Out[2]:

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.li
0	1	debt_consolidation	0.1189	829.10	11.350407	19.48	737	5639.9583
1	1	credit_card	0.1071	228.22	11.082143	14.29	707	2760.0000
2	1	debt_consolidation	0.1357	366.86	10.373491	11.63	682	4710.0000
3	1	debt_consolidation	0.1008	162.34	11.350407	8.10	712	2699.9583
4	1	credit_card	0.1426	102.92	11.299732	14.97	667	4066.0000

In [3]:
```
# data summary 1
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   credit.policy      9578 non-null   int64  
 1   purpose            9578 non-null   object 
 2   int.rate           9578 non-null   float64
 3   installment        9578 non-null   float64
 4   log.annual.inc     9578 non-null   float64
 5   dti                9578 non-null   float64
 6   fico               9578 non-null   int64  
 7   days.with.cr.line  9578 non-null   float64
 8   revol.bal          9578 non-null   int64  
 9   revol.util         9578 non-null   float64
 10  inq.last.6mths     9578 non-null   int64  
 11  delinq.2yrs        9578 non-null   int64  
 12  pub.rec            9578 non-null   int64  
 13  not.fully.paid     9578 non-null   int64  
dtypes: float64(6), int64(7), object(1)
memory usage: 1.0+ MB
```

In [4]:
```
# statistical summary
df.describe().astype(float).round(2)
```

Out[4]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	rev
count	9578.0	9578.00	9578.00	9578.00	9578.00	9578.00	9578.00	9
mean	0.8	0.12	319.09	10.93	12.61	710.85	4560.77	16
std	0.4	0.03	207.07	0.61	6.88	37.97	2496.93	33
min	0.0	0.06	15.67	7.55	0.00	612.00	178.96	
25%	1.0	0.10	163.77	10.56	7.21	682.00	2820.00	3
50%	1.0	0.12	268.95	10.93	12.66	707.00	4139.96	8
75%	1.0	0.14	432.76	11.29	17.95	737.00	5730.00	18
max	1.0	0.22	940.14	14.53	29.96	827.00	17639.96	1207

In [5]:
```
# missing values
missing_data = df.isnull().sum()
missing_data

# missing_data = missing_data[missing_data > 0]
# missing_data # will return all the columns with missing values as series.
```

Out[5]:
```
credit.policy        0
purpose              0
int.rate             0
installment          0
log.annual.inc       0
dti                  0
fico                 0
days.with.cr.line    0
revol.bal            0
revol.util           0
inq.last.6mths       0
delinq.2yrs          0
pub.rec              0
not.fully.paid       0
dtype: int64
```

Pre-Processing

In [6]:
```
# making 'interest rate' column more meaningful
df['int.rate'] = df['int.rate'] * 100
```

In [7]:
```
df=df.round(2)
df
```

Out[7]:

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with
0	1	debt_consolidation	11.89	829.10	11.35	19.48	737	
1	1	credit_card	10.71	228.22	11.08	14.29	707	
2	1	debt_consolidation	13.57	366.86	10.37	11.63	682	
3	1	debt_consolidation	10.08	162.34	11.35	8.10	712	
4	1	credit_card	14.26	102.92	11.30	14.97	667	
...	...	...	...	...	...	...	...	
9573	0	all_other	14.61	344.76	12.18	10.39	672	1
9574	0	all_other	12.53	257.70	11.14	0.21	722	
9575	0	debt_consolidation	10.71	97.81	10.60	13.09	687	
9576	0	home_improvement	16.00	351.58	10.82	19.18	692	
9577	0	debt_consolidation	13.92	853.43	11.26	16.28	732	

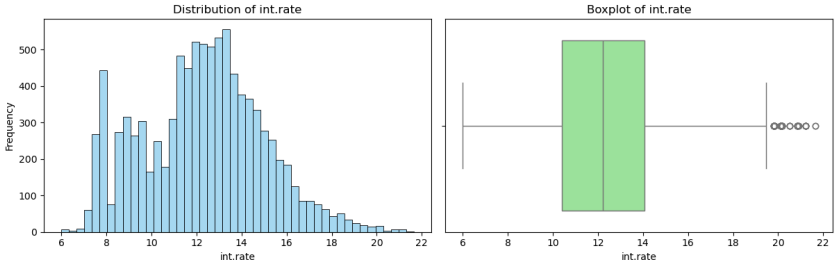
9578 rows × 14 columns

Individual feature review

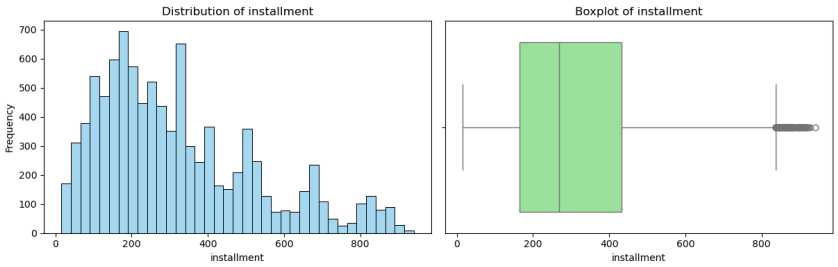
In [8]:
```
num_features = ['purpose', 'int.rate', 'installment', 'log.annual.inc','dti', 'fico', '
non_num_features = ['credit.policy','not.fully.paid']
```

In [9]:
```
# Seperating Numerical columns from Binary columns
non_boolean_numerical_features = ['int.rate', 'installment', 'log.annual.inc', 'dti', '
                                  'days.with.cr.line', 'revol.bal', 'revol.util', 'inq.
                                  'delinq.2yrs', 'pub.rec']

# Seperating binary data columns
boolean_numeric_features = ['credit.policy', 'not.fully.paid']

# Visualize the distributions and box plots for numerical features, including log-transf

# iterate through each column of 'non_boolen_numerical_feature'
for column in non_boolean_numerical_features:
    fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(18, 4))

    # Histogram for the distribution
    # print histogram of values distribution of each column
    sns.histplot(df[column], kde=False, color='skyblue', ax=ax1)
    ax1.set_title(f'Distribution of {column}')
    ax1.set_ylabel('Frequency')

    # Boxplot for the variable
    # boxplot to understand outliers or distribution
    sns.boxplot(x=df[column], color='lightgreen', ax=ax2)
    ax2.set_title(f'Boxplot of {column}')
```

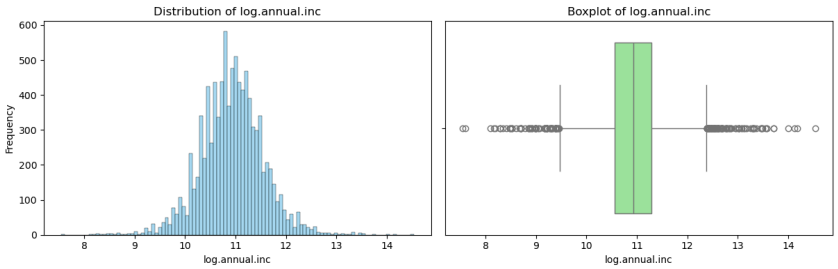
```
    # Log transformation and plot if the data is skewed
    # if data is skewed for more than 1 we'll transform the distribution.
    if df[column].skew() > 1:
        df[column+'_log'] = np.log1p(df[column])
        sns.histplot(df[column+'_log'], kde=False, color='orange', ax=ax3)
        ax3.set_title(f'Log-transformed Distribution of {column}')
    else:
        ax3.set_title(f'Log-transformed plot not necessary for {column}')
        ax3.axis('off')

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



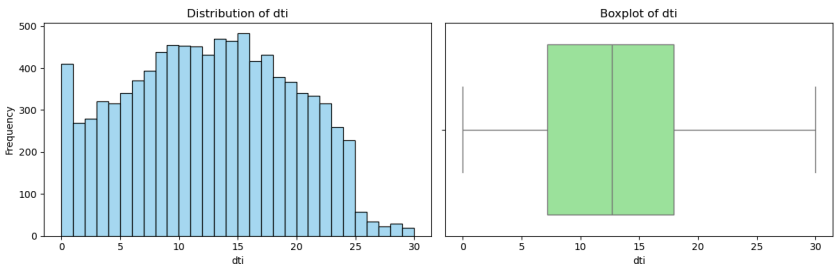
Log-transformed plot not necessary for int.rate



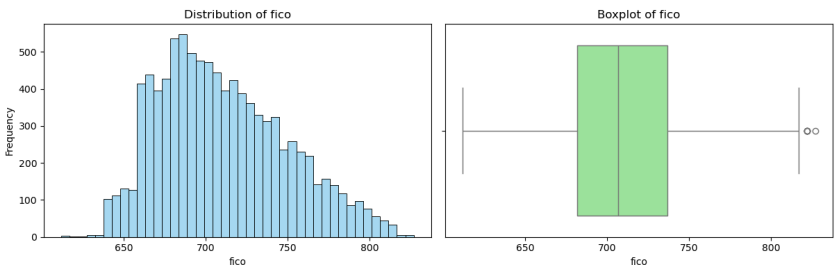
Log-transformed plot not necessary for installment



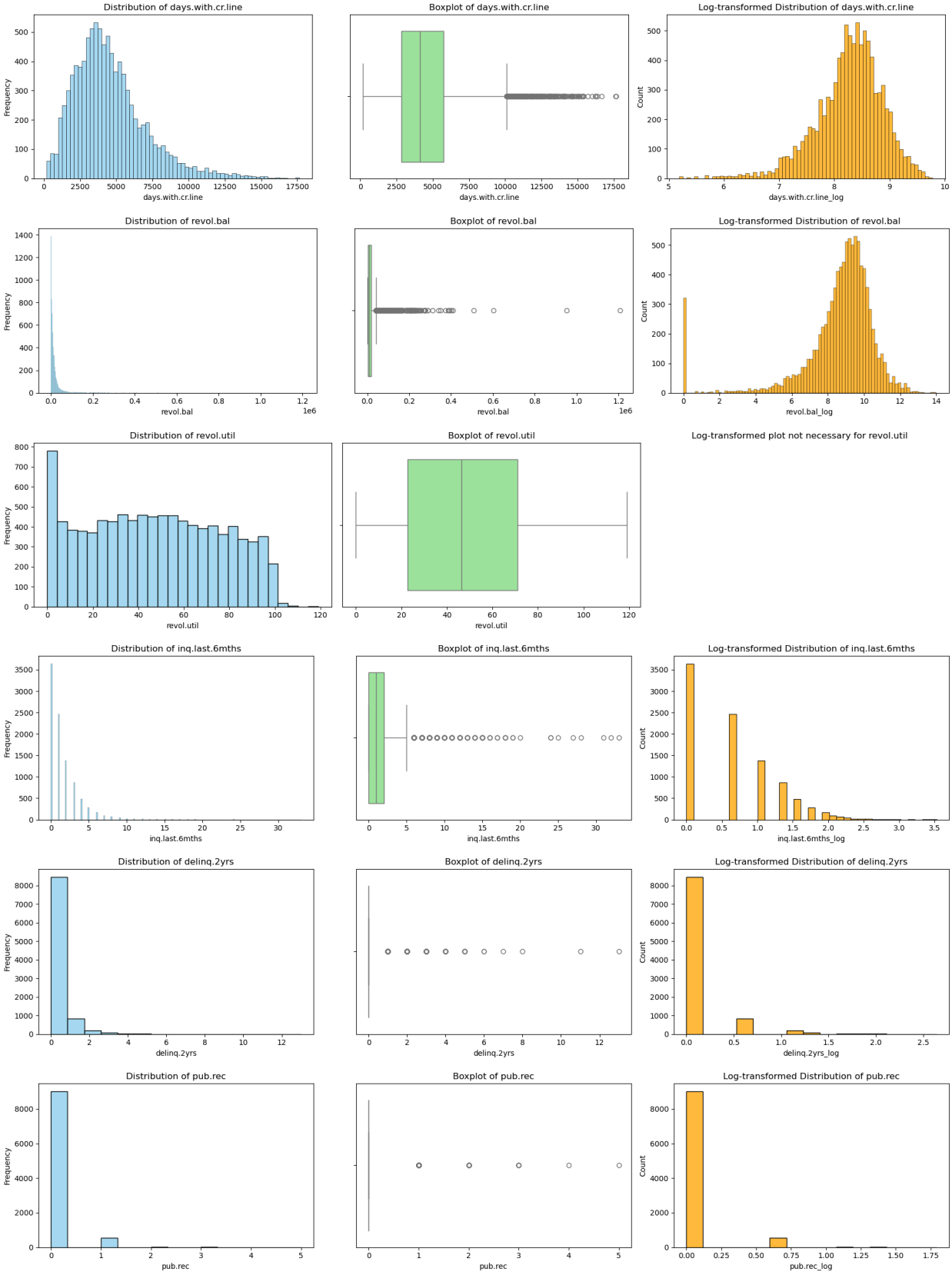
Log-transformed plot not necessary for log.annual.inc



Log-transformed plot not necessary for dti



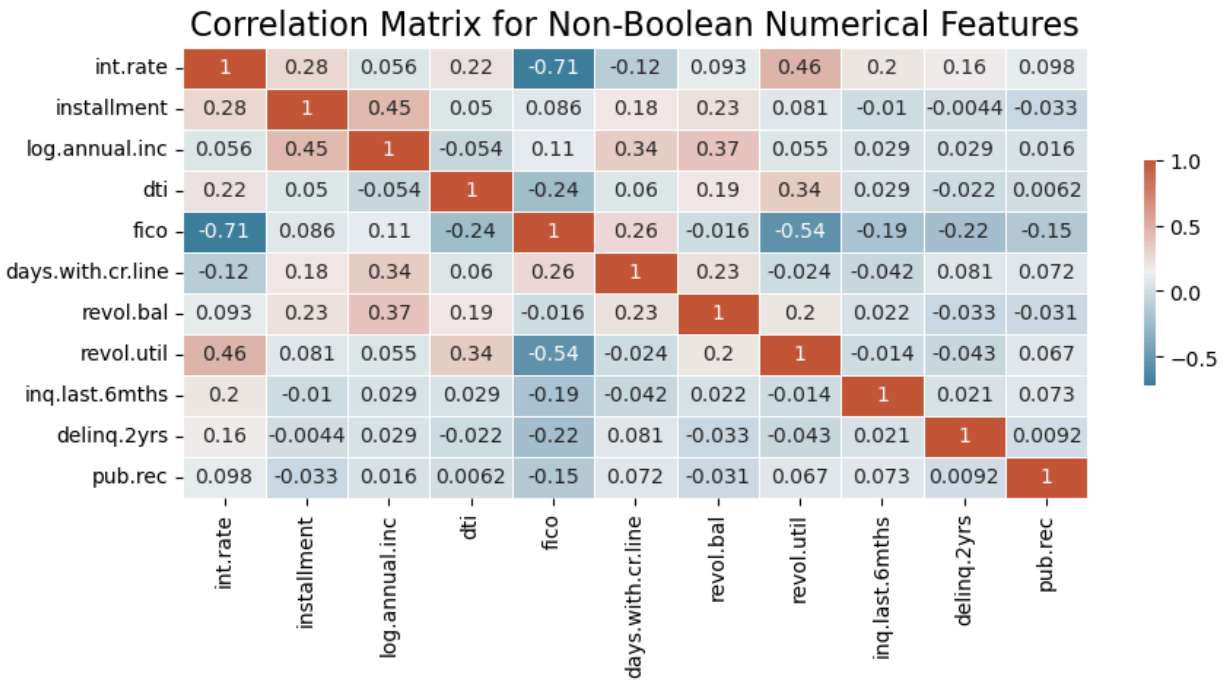
Log-transformed plot not necessary for fico



This single feature analysis helps us to understand about Distribution of data, Outliers in feature and Skewness in features.

Infer Feature Relation

In [14]:
```
# Correletion matrix for columns in dataframe
corr_matrix = df[non_boolean_numerical_features].corr()
corr_matrix # this is corelation of column by column.

# set up matplotlib figure
plt.figure(figsize=(10,4))

# generate the custom diverging color map
cmap  = sns.diverging_palette(230, 20, as_cmap=True)

# plotting heatmap
sns.heatmap(corr_matrix,annot=True,cmap=cmap,linewidth=.5,cbar_kws={"shrink": .5})

plt.title('Correlation Matrix for Non-Boolean Numerical Features', fontsize=16)
plt.show()
```



This multiple correlation analysis helps us to understand how closely/far 2 features are co-related to each other.

EDA

In [86]:
```
# Average interest for each 'purpose' for each purpose.
avg_int = pd.DataFrame(df.groupby(['purpose'])['int.rate'].mean().sort_values().round(2
avg_int.rename({'int.rate' : 'avg_int_rate'},inplace=True,axis=1)
avg_int
```

Out[86]:

	purpose	avg_int_rate
0	major_purchase	11.42
1	all_other	11.68
2	home_improvement	11.75
3	credit_card	11.97
4	educational	11.99
5	debt_consolidation	12.66
6	small_business	13.81

This information shows that 'small business' and 'education' are having highest average interest in market.

In [ ]:
```
# finding total unpaid debt by the end of every credit cycle
unpaid_amt = pd.DataFrame(df.groupby('purpose')['revol.bal'].sum().sort_values(ascending
min_unpaid_amt = pd.DataFrame(df.groupby('purpose')['revol.bal'].min().sort_values(asce
max_unpaid_amt = pd.DataFrame(df.groupby('purpose')['revol.bal'].max().sort_values(asce


# joining
unpaid_amt = pd.merge(unpaid_amt,min_unpaid_amt,how='left',on='purpose')
unpaid_amt = pd.merge(unpaid_amt,max_unpaid_amt,how='left',on='purpose')
unpaid_amt.rename({'revol.bal_x' : 'total_unpaid_amt','revol.bal_y' : 'min_unpaid_amt',
unpaid_amt
```

Out[ ]:

	purpose	total_unpaid_amt	min_unpaid_amt	max_unpaid_amt
0	debt_consolidation	67849534	0	290341
1	all_other	30030366	0	602519
2	credit_card	29253186	0	394107
3	small_business	17072765	0	1207359
4	home_improvement	10899788	0	311616
5	educational	3714312	0	226567
6	major_purchase	3181995	0	111115

This numbers shows us total amount that is being unpaid at end of each credit cycle for each purpose. Also this results shows Minimum and Maximum pending amount in each 'Purpose' that is unpaid at the end of every credit score. This helps us to understand High risk loan purposes and potential risk in each purpose.

In [92]:
```
# Loan Approval Rate Based on Credit Policy
approval_rate = df.groupby('credit.policy')['not.fully.paid'].mean()
print(approval_rate)
```

```
credit.policy
0    0.277837
1    0.131518
Name: not.fully.paid, dtype: float64
```

In [93]:
```
# Loan Purpose and Default Rates
purpose_default_rate = df.groupby('purpose')['not.fully.paid'].mean()
print(purpose_default_rate)
```

```
purpose
all_other            0.166023
credit_card          0.115689
debt_consolidation   0.152388
educational          0.201166
home_improvement     0.170111
major_purchase       0.112128
small_business       0.277868
Name: not.fully.paid, dtype: float64
```

In [94]:
```
# Debt-to-Income Ratio (dti) and Default Risk
dti_default = df[df['not.fully.paid'] == 1]['dti'].describe()
print(dti_default)
```

```
count    1533.000000
mean       13.195838
std         7.006769
min         0.000000
25%         7.830000
50%        13.340000
75%        18.830000
max        29.960000
Name: dti, dtype: float64
```

In [96]:
```
# FICO Score (fico) and Loan Default
fico_default = df[df['fico'] < 650].groupby('not.fully.paid').size()
print(fico_default)
```

```
not.fully.paid
0    157
1     74
dtype: int64
```

In [97]:
```
# Income vs. Default Risk (Log Annual Income)
income_default = df.groupby('log.annual.inc')['not.fully.paid'].mean()
print(income_default)
```

```
log.annual.inc
7.55     0.0
7.60     1.0
8.10     0.0
8.16     0.0
8.19     1.0
        ...
13.71    0.0
14.00    0.0
14.12    0.0
14.18    0.0
14.53    0.0
Name: not.fully.paid, Length: 376, dtype: float64
```

In [98]:
```
# Installment Amount vs. Default Risk
installment_default = df.groupby('installment')['not.fully.paid'].mean()
print(installment_default)
```

```
installment
15.67     0.0
15.69     0.0
15.75     0.0
15.76     0.0
15.91     1.0
         ...
916.95    0.0
918.02    1.0
922.42    0.0
926.83    0.5
940.14    0.0
Name: not.fully.paid, Length: 4788, dtype: float64
```