In [ ]:
```
import pandas as pd
pd.set_option("display.max_columns", None)
import seaborn as sns
import matplotlib.pyplot as plt
```

EDA and Data Processing

In [ ]:
```
df = pd.read_csv("dataset/logistic_regression.csv")
df.head()
```

Out[ ]:

	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	earl
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	Not Verified	Jan-2015	Fully Paid	vacation	Vacation	26.24	
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	Not Verified	Jan-2015	Fully Paid	debt_consolidation	Debt consolidation	22.05	
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	Source Verified	Jan-2015	Fully Paid	credit_card	Credit card refinancing	12.79	
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 years	RENT	54000.0	Not Verified	Nov-2014	Fully Paid	credit_card	Credit card refinancing	2.60	
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years	MORTGAGE	55000.0	Verified	Apr-2013	Charged Off	credit_card	Credit Card Refinance	33.95	

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

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

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

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

In [ ]:
```
df.loan_status.value_counts()
```

Out[ ]:
```
loan_status
Fully Paid     318357
Charged Off     77673
Name: count, dtype: int64
```

term, emp_length to convert to Float/Int type

In [ ]:
```
print(df["term"].unique())
df["term"] = pd.to_numeric(df["term"].apply(lambda x: x.strip().split(" ")[0]))
```

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

In [ ]:
```
print(df["emp_length"].unique())
def get_emp_duration_type(value):
    return (
        "greaterthan" if pd.notna(value) and list(value.strip())[2] == "+"
        else "lessthan" if pd.notna(value) and list(value.strip())[0] == "<"
        else "equalsto"
        )

df["emp_duration_type"] = df["emp_length"].apply(get_emp_duration_type)
```

```
['10+ years' '4 years' '< 1 year' '6 years' '9 years' '2 years' '3 years'
 '8 years' '7 years' '5 years' '1 year' nan]
```

In [ ]:
```
df["emp_duration_type"].value_counts()
```

Out[ ]:
```
emp_duration_type
equalsto       238264
greaterthan    126041
lessthan        31725
Name: count, dtype: int64
```

In [ ]:
```
print(df["emp_length"].unique())
def get_emp_duration(value):
    return (
        10 if pd.notna(value) and list(value.strip())[2] == "+"
        else 1 if pd.notna(value) and list(value.strip())[0] == "<"
        else int(value.strip().split()[0]) if pd.notna(value)
        else value
        )

df["emp_length"] = df["emp_length"].apply(get_emp_duration)
```

```
['10+ years' '4 years' '< 1 year' '6 years' '9 years' '2 years' '3 years'
 '8 years' '7 years' '5 years' '1 year' nan]
```

In [ ]:
```
df["emp_length"].value_counts()
```

Out[ ]:
```
emp_length
10.0    126041
1.0      57607
2.0      35827
3.0      31665
5.0      26495
4.0      23952
6.0      20841
7.0      20819
8.0      19168
9.0      15314
Name: count, dtype: int64
```

issue_d, earliest_cr_line to convert to Datetime

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

Out[ ]:

	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	earlie
0	10000.0	36	11.44	329.48	B	B4	Marketing	10.0	RENT	117000.0	Not Verified	Jan-2015	Fully Paid	vacation	Vacation	26.24	
1	8000.0	36	11.99	265.68	B	B5	Credit analyst	4.0	MORTGAGE	65000.0	Not Verified	Jan-2015	Fully Paid	debt_consolidation	Debt consolidation	22.05	
2	15600.0	36	10.49	506.97	B	B3	Statistician	1.0	RENT	43057.0	Source Verified	Jan-2015	Fully Paid	credit_card	Credit card refinancing	12.79	
3	7200.0	36	6.49	220.65	A	A2	Client Advocate	6.0	RENT	54000.0	Not Verified	Nov-2014	Fully Paid	credit_card	Credit card refinancing	2.60	
4	24375.0	60	17.27	609.33	C	C5	Destiny Management Inc.	9.0	MORTGAGE	55000.0	Verified	Apr-2013	Charged Off	credit_card	Credit Card Refinance	33.95	

In [ ]:
```
print(df["issue_d"].unique())
df["issue_month"] = df["issue_d"].apply(lambda x: x.strip().split("-")[0])
df["issue_year"] = df["issue_d"].apply(lambda x: x.strip().split("-")[1])
```

```
['Jan-2015' 'Nov-2014' 'Apr-2013' 'Sep-2015' 'Sep-2012' 'Oct-2014'
 'Apr-2012' 'Jun-2013' 'May-2014' 'Dec-2015' 'Apr-2015' 'Oct-2012'
 'Jul-2014' 'Feb-2013' 'Oct-2015' 'Jan-2014' 'Mar-2016' 'Apr-2014'
 'Jun-2011' 'Apr-2010' 'Jun-2014' 'Oct-2013' 'May-2013' 'Feb-2015'
 'Oct-2011' 'Jun-2015' 'Aug-2013' 'Feb-2014' 'Dec-2011' 'Mar-2013'
 'Jun-2016' 'Mar-2014' 'Nov-2013' 'Dec-2014' 'Apr-2016' 'Sep-2013'
 'May-2016' 'Jul-2015' 'Jul-2013' 'Aug-2014' 'May-2008' 'Mar-2010'
 'Dec-2013' 'Mar-2012' 'Mar-2015' 'Sep-2011' 'Jul-2012' 'Dec-2012'
 'Sep-2014' 'Nov-2012' 'Nov-2015' 'Jan-2011' 'May-2012' 'Feb-2016'
 'Jun-2012' 'Aug-2012' 'Jan-2016' 'May-2015' 'Oct-2016' 'Aug-2015'
 'Jul-2016' 'May-2009' 'Aug-2016' 'Jan-2012' 'Jan-2013' 'Nov-2010'
 'Jul-2011' 'Mar-2011' 'Feb-2012' 'May-2011' 'Aug-2010' 'Nov-2016'
 'Jul-2010' 'Sep-2010' 'Dec-2010' 'Feb-2011' 'Jun-2009' 'Aug-2011'
 'Dec-2016' 'Mar-2009' 'Jun-2010' 'May-2010' 'Nov-2011' 'Sep-2016'
 'Oct-2009' 'Mar-2008' 'Nov-2008' 'Dec-2009' 'Oct-2010' 'Sep-2009'
 'Oct-2007' 'Aug-2009' 'Jul-2009' 'Nov-2009' 'Jan-2010' 'Dec-2008'
 'Feb-2009' 'Oct-2008' 'Apr-2009' 'Feb-2010' 'Apr-2011' 'Apr-2008'
 'Aug-2008' 'Jan-2009' 'Feb-2008' 'Aug-2007' 'Sep-2008' 'Dec-2007'
 'Jan-2008' 'Sep-2007' 'Jun-2008' 'Jul-2008' 'Jun-2007' 'Nov-2007'
 'Jul-2007']
```

In [ ]:
```
df.issue_month.value_counts()
```

Out[ ]:
```
issue_month
Oct    42130
Jul    39714
Jan    34682
Nov    34068
Apr    33223
Aug    32816
Mar    31919
May    31895
Jun    30140
Dec    29082
Feb    28742
Sep    27619
Name: count, dtype: int64
```

In [ ]:
```
df.issue_year.value_counts()
```

Out[ ]:
```
issue_year
2014    102860
2013     97662
2015     94264
2012     41202
2016     28088
2011     17435
2010      9258
2009      3826
2008      1240
2007       195
Name: count, dtype: int64
```

In [ ]:
```
df["earliest_cr_line"].nunique()
```

Out[ ]:
```
684
```

In [ ]:
```
print(df["earliest_cr_line"].unique())
df["earliest_cr_month"] = df["earliest_cr_line"].apply(lambda x: x.strip().split("-")[0])
df["earliest_cr_year"] = df["earliest_cr_line"].apply(lambda x: x.strip().split("-")[1])
```

```
['Jun-1990' 'Jul-2004' 'Aug-2007' 'Sep-2006' 'Mar-1999' 'Jan-2005'
 'Aug-2005' 'Sep-1994' 'Jun-1994' 'Dec-1997' 'Dec-1990' 'May-1984'
 'Apr-1995' 'Jan-1997' 'May-2001' 'Mar-1982' 'Sep-1996' 'Jan-1990'
 'Mar-2000' 'Jan-2006' 'Oct-2006' 'Jan-2003' 'May-2008' 'Oct-2003'
 'Jun-2004' 'Jan-1999' 'Apr-1994' 'Apr-1998' 'Jul-2007' 'Apr-2002'
 'Oct-2007' 'Jun-2009' 'May-1997' 'Jul-2006' 'Sep-2003' 'Aug-1992'
 'Dec-1988' 'Feb-2002' 'Jan-1992' 'Aug-2001' 'Dec-2010' 'Oct-1999'
 'Sep-2004' 'Aug-1994' 'Jul-2003' 'Apr-2000' 'Dec-2004' 'Jun-1995'
 'Dec-2003' 'Jul-1994' 'Oct-1990' 'Dec-2001' 'Apr-1999' 'Feb-1995'
 'May-2003' 'Oct-2002' 'Mar-2004' 'Aug-2003' 'Oct-2000' 'Nov-2004'
 'Mar-2010' 'Mar-1996' 'May-1994' 'Jun-1996' 'Nov-1986' 'Jan-2001'
 'Jan-2002' 'Mar-2001' 'Sep-2012' 'Apr-2006' 'May-1998' 'Dec-2002'
 'Nov-2003' 'Oct-2005' 'May-1990' 'Jun-2003' 'Jun-2001' 'Jan-1998'
 'Oct-1978' 'Feb-2001' 'Jun-2006' 'Aug-1993' 'Apr-2001' 'Nov-2001'
 'Feb-2003' 'Jun-1993' 'Sep-1992' 'Nov-1992' 'Jun-1983' 'Oct-2001'
 'Jul-1999' 'Sep-1997' 'Nov-1993' 'Feb-1993' 'Apr-2007' 'Nov-1999'
 'Nov-2005' 'Dec-1992' 'Mar-1986' 'May-1989' 'Dec-2000' 'Mar-1991'
 'Mar-2005' 'Jun-2010' 'Dec-1998' 'Sep-2001' 'Nov-2000' 'Jan-1994'
 'Aug-2002' 'Jan-2011' 'Aug-2008' 'Jun-2005' 'Nov-1997' 'May-1996'
 'Apr-2010' 'May-1993' 'Sep-2005' 'Jun-1992' 'Apr-1986' 'Aug-1996'
 'Aug-1997' 'Jul-2005' 'May-2011' 'Sep-2002' 'Jan-1989' 'Aug-1999'
 'Feb-1992' 'Sep-1999' 'Jul-2001' 'May-1980' 'Oct-2008' 'Nov-2007'
 'Apr-1997' 'Jun-1986' 'Sep-1998' 'Jun-1982' 'Oct-1981' 'Feb-1994'
 'Dec-1984' 'Nov-1991' 'Nov-2006' 'Aug-2000' 'Oct-2004' 'Jun-2011'
 'Apr-1988' 'May-2004' 'Aug-1988' 'Mar-1994' 'Aug-2004' 'Dec-2006'
 'Nov-1998' 'Oct-1997' 'Mar-1989' 'Feb-1988' 'Jul-1982' 'Nov-1995'
 'Mar-1997' 'Oct-1994' 'Jul-1998' 'Jun-2002' 'May-1991' 'Oct-2011'
 'Sep-2007' 'Jan-2007' 'Jan-2010' 'Mar-1987' 'Feb-1997' 'Oct-1986'
 'Mar-2002' 'Jul-1993' 'Mar-2007' 'Aug-1989' 'Oct-1995' 'May-2007'
 'Dec-1993' 'Jun-1989' 'Apr-2004' 'Jun-1997' 'Apr-1996' 'Apr-1992'
 'Oct-1998' 'Mar-1983' 'Mar-1985' 'Oct-1993' 'Feb-2000' 'Apr-2003'
 'Oct-1985' 'Jul-1985' 'May-1978' 'Sep-2010' 'Oct-1996' 'Sep-2009'
 'Jun-1999' 'Jan-2000' 'Sep-1987' 'Aug-1998' 'Jan-1995' 'Jul-1988'
 'May-2000' 'Jun-1981' 'Feb-1998' 'Nov-1996' 'Aug-1967' 'Dec-1999'
 'Aug-2006' 'Nov-2009' 'Jul-2000' 'Mar-1988' 'Jul-1992' 'Jul-1991'
 'Mar-1990' 'May-1986' 'Jun-1991' 'Dec-1987' 'Jul-1996' 'Jul-1997'
 'Aug-1990' 'Jan-1988' 'Dec-2005' 'Mar-2003' 'Feb-1999' 'Nov-1990'
 'Jun-2000' 'Dec-1996' 'Jan-2004' 'May-1999' 'Sep-1972' 'Jul-1981'
 'Sep-1993' 'Feb-2009' 'Nov-2002' 'Nov-1969' 'Jan-1993' 'May-2005'
 'Sep-1982' 'Apr-1990' 'Feb-1996' 'Mar-1993' 'Apr-1978' 'Jul-1995'
 'May-1995' 'Apr-1991' 'Mar-1998' 'Aug-1991' 'Jul-2002' 'Oct-1989'
 'Apr-1984' 'Dec-2009' 'Sep-2000' 'Jan-1982' 'Jun-1998' 'Jan-1996'
 'Nov-1987' 'May-2010' 'Jul-1989' 'Jun-1987' 'Oct-1987' 'Aug-1995'
 'Feb-2004' 'Oct-1991' 'Dec-1989' 'Oct-1992' 'Feb-2005' 'Apr-1993'
 'Dec-1985' 'Sep-1979' 'Feb-2007' 'Nov-1989' 'Apr-2005' 'Mar-1978'
 'Sep-1985' 'Nov-1994' 'Jun-2008' 'Apr-1987' 'Dec-1983' 'Dec-2007'
 'May-1979' 'May-1992' 'Jul-1990' 'Mar-1995' 'Feb-2006' 'Feb-1985'
 'Sep-1989' 'Aug-2009' 'Nov-2008' 'Nov-1981' 'Jan-2008' 'Aug-1987'
 'Nov-1985' 'Dec-1965' 'Sep-1995' 'Jan-1986' 'Oct-2009' 'May-2002'
 'Aug-1980' 'Sep-1977' 'Sep-1988' 'Oct-1984' 'May-1988' 'Aug-1984'
 'Nov-1988' 'May-1974' 'Nov-1982' 'Oct-1983' 'Sep-1991' 'Feb-1984'
 'Feb-1991' 'Jan-1981' 'Jun-1985' 'Dec-1976' 'Dec-1994' 'Dec-1980'
 'Sep-1984' 'Jun-2007' 'Aug-1979' 'Sep-2008' 'Apr-1983' 'Mar-2006'
 'Jun-1984' 'Jul-1984' 'Jan-1985' 'Dec-1995' 'Apr-2008' 'Mar-2008'
 'Jan-1983' 'Dec-1986' 'Jun-1979' 'Dec-1975' 'Nov-1983' 'Jul-1986'
 'Nov-1977' 'Dec-1982' 'May-1985' 'Feb-1983' 'Aug-1982' 'Oct-1980'
 'Mar-1979' 'Jan-1978' 'Mar-1984' 'May-1983' 'Jul-2008' 'Apr-1982'
 'Jul-1983' 'Feb-1990' 'Dec-2008' 'Jul-1975' 'Dec-1971' 'Feb-2008'
 'Mar-2011' 'Feb-1987' 'Feb-1989' 'Aug-1985' 'Jul-2010' 'Apr-1989'
 'Feb-1980' 'May-2006' 'Nov-2010' 'Apr-2009' 'Feb-2010' 'May-1976'
 'Feb-1981' 'Jan-2012' 'Oct-1988' 'Nov-1984' 'May-1982' 'Oct-1975'
 'Jun-1988' 'May-1972' 'Apr-2013' 'Sep-1990' 'Oct-1982' 'Feb-2013'
 'Mar-1992' 'Aug-1981' 'Feb-2011' 'Nov-1974' 'Feb-1978' 'Sep-1983'
 'Jul-2011' 'Nov-1979' 'Aug-1983' 'Apr-1985' 'Jul-2009' 'Jan-1971'
 'Jul-1987' 'Aug-1978' 'Aug-2010' 'Oct-1976' 'Aug-1986' 'Jan-1991'
 'Dec-1991' 'May-2009' 'Aug-2011' 'Jun-1964' 'Jan-1974' 'May-1981'
 'Jun-1972' 'Jun-1978' 'Sep-1986' 'Jan-1987' 'Jan-1975' 'Feb-1982'
 'Jan-1980' 'Feb-1977' 'Sep-1980' 'Nov-1978' 'Jul-1974' 'Jun-1970'
 'Jan-1984' 'Nov-1980' 'May-1987' 'Sep-1970' 'Jan-1976' 'Feb-1986'
 'Oct-2010' 'Apr-1979' 'Oct-1979' 'Jan-1979' 'Sep-2011' 'Jul-1979'
 'Sep-1975' 'Mar-1981' 'Aug-1971' 'Apr-1980' 'Apr-1977' 'Jan-1965'
 'Nov-1976' 'Nov-1970' 'Nov-2011' 'Nov-1973' 'Sep-1981' 'Jul-1980'
 'Mar-2012' 'Dec-1974' 'Mar-1977' 'Dec-1977' 'May-2012' 'Dec-1979'
 'Jan-2009' 'Jan-1970' 'Dec-2011' 'Feb-1979' 'Mar-1976' 'Jan-1973'
 'Oct-1973' 'Mar-1969' 'Oct-1977' 'Mar-1975' 'Aug-1977' 'Jun-1969'
 'Oct-1963' 'Nov-1960' 'Aug-1970' 'Feb-1975' 'Sep-1974' 'May-1966'
 'Apr-1972' 'Apr-1973' 'Apr-2012' 'May-1975' 'Sep-1966' 'Feb-1969'
 'Feb-2012' 'Jan-1961' 'Aug-1973' 'Feb-1972' 'Apr-1975' 'Jul-1978'
 'Oct-1970' 'Mar-1980' 'Sep-1976' 'Apr-2011' 'Nov-2012' 'Aug-1976'
 'Jun-1975' 'Apr-1981' 'Mar-2009' 'Jun-1977' 'Apr-1971' 'Sep-1969'
 'Jun-2012' 'Apr-1976' 'Feb-1965' 'Jul-1977' 'Jun-1976' 'Mar-1973'
 'Oct-1972' 'Dec-1978' 'Nov-1967' 'Sep-1967' 'Nov-1971' 'Jun-1980'
 'May-1964' 'Feb-1971' 'May-1970' 'Apr-1970' 'Mar-1971' 'Apr-1969'
 'Jan-1963' 'Jun-1974' 'Oct-1974' 'May-1977' 'Dec-1981' 'Jan-1969'
 'Feb-1976' 'Mar-1970' 'Aug-1968' 'Feb-1970' 'Jun-1971' 'Jun-1963'
 'Jun-2013' 'Mar-1972' 'Aug-2012' 'Jan-1967' 'Feb-1968' 'Dec-1969'
 'Jan-1977' 'Jul-1970' 'Feb-1973' 'Mar-1974' 'Feb-1974' 'Dec-1960'
 'Jul-1972' 'Jul-1973' 'Sep-1964' 'Jul-1965' 'Oct-1958' 'Jul-2012'
 'Jun-1973' 'Sep-1978' 'Nov-1975' 'Jul-1963' 'Jan-1964' 'Dec-1968'
 'May-1958' 'Sep-1973' 'May-1971' 'Dec-1972' 'Aug-1965' 'Jul-1976'
 'Oct-2012' 'May-1973' 'Apr-1955' 'Apr-1966' 'Jan-1968' 'Nov-1968'
 'Oct-1969' 'Mar-2013' 'Jan-2013' 'Jul-1967' 'Oct-1965' 'Jan-1966'
 'Aug-1972' 'Jul-1969' 'May-1965' 'Jan-1953' 'Aug-1974' 'May-1968'
 'Aug-1969' 'May-2013' 'Oct-1967' 'Aug-1975' 'Apr-1974' 'Sep-1971'
 'Apr-1968' 'Jul-1971' 'Jan-1972' 'Nov-1965' 'Dec-1970' 'Dec-1973'
 'Nov-1972' 'Oct-1959' 'Oct-1962' 'Apr-1967' 'Oct-1971' 'Nov-1963'
 'Oct-1968' 'Dec-1962' 'Jun-1960' 'Jan-1960' 'Sep-2013' 'May-1969'
 'Dec-1966' 'Feb-1967' 'Dec-1967' 'Aug-1961' 'Sep-1968' 'Oct-1964'
 'Aug-1966' 'Jul-1966' 'Apr-1964' 'Sep-1962' 'Jul-2013' 'Jun-1967'
 'Apr-1965' 'Jun-1966' 'Jan-1955' 'Jan-1962' 'Feb-1964' 'Aug-1958'
 'Jul-1968' 'May-1967' 'Dec-1959' 'Sep-1963' 'Dec-2012' 'Dec-1963'
 'Jan-1944' 'Jun-1965' 'May-1962' 'Mar-1967' 'Mar-1968' 'Jan-1956'
 'Sep-1965' 'Dec-1951' 'Aug-2013' 'Jun-1968' 'Mar-1965' 'Oct-1957'
 'Nov-1966' 'Dec-1958' 'Feb-1957' 'Feb-1963' 'Mar-1963' 'Jan-1959'
 'May-1955' 'Feb-1966' 'Nov-1950' 'Mar-1964' 'Jan-1958' 'Nov-1964'
 'Sep-1961' 'Apr-1963' 'Jul-1964' 'Nov-1955' 'Jun-1957' 'Dec-1964'
 'Nov-1953' 'Apr-1961' 'Mar-1966' 'Oct-1960' 'Jul-1959' 'Jul-1961'
 'Jan-1954' 'Dec-1956' 'Mar-1962' 'Jul-1960' 'Sep-1959' 'Dec-1950'
 'Oct-1966' 'Apr-1960' 'Jul-1958' 'Nov-1954' 'Nov-1957' 'Jun-1962'
 'May-1963' 'Jul-1955' 'Oct-1950' 'Dec-1961' 'Aug-1951' 'Oct-2013'
 'Aug-1964' 'Apr-1962' 'Jun-1955' 'Jul-1962' 'Jan-1957' 'Nov-1958'
 'Jul-1951' 'Nov-1959' 'Apr-1958' 'Mar-1960' 'Sep-1957' 'Nov-1961'
 'Sep-1960' 'May-1959' 'Jun-1959' 'Feb-1962' 'Sep-1956' 'Aug-1960'
 'Feb-1961' 'Jan-1948' 'Aug-1963' 'Oct-1961' 'Aug-1962' 'Aug-1959']
```

In [ ]:
```
df.earliest_cr_month.value_counts()
```

Out[ ]:
```
earliest_cr_month
Oct    38291
Sep    37673
Aug    37349
Nov    35583
Dec    33687
Jul    31972
Mar    31617
Jan    30694
Jun    30445
May    30445
Apr    29231
Feb    29043
Name: count, dtype: int64
```

In [ ]:
```
df.earliest_cr_year.value_counts()
```

Out[ ]:
```
earliest_cr_year
2000    29366
2001    29083
1999    26491
2002    25901
2003    23657
        ...
1951        3
1950        3
1953        2
1944        1
1948        1
Name: count, Length: 65, dtype: int64
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 32 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   loan_amnt             396030 non-null  float64
 1   term                  396030 non-null  int64  
 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  float64
 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 
 27  emp_duration_type     396030 non-null  object 
 28  issue_month           396030 non-null  object 
 29  issue_year            396030 non-null  object 
 30  earliest_cr_month     396030 non-null  object 
 31  earliest_cr_year      396030 non-null  object 
dtypes: float64(13), int64(1), object(18)
memory usage: 96.7+ MB
```

In [ ]:
```
df["address"][:5]
```

Out[ ]:
```
0       0174 Michelle Gateway\r\nMendozaberg, OK 22690
1    1076 Carney Fort Apt. 347\r\nLoganmouth, SD 05113
2    87025 Mark Dale Apt. 269\r\nNew Sabrina, WV 05113
3              823 Reid Ford\r\nDelacruzside, MA 00813
4               679 Luna Roads\r\nGreggshire, VA 11650
Name: address, dtype: object
```

In [ ]:
```
import re

def clean_string(input_string):
    # Remove special characters
    cleaned_string = re.sub(r'[^A-Za-z0-9\s]', '', input_string)

    # Replace multiple spaces with a single space
    cleaned_string = re.sub(r'\s+', ' ', cleaned_string).strip()

    return cleaned_string

df["city"] = df["address"].apply(lambda x : clean_string(" ".join(x.split("\r\n")[-1].split(" ")[:-2])))
df["state"] = df["address"].apply(lambda x : x.split("\r\n")[-1].split(" ")[-2])
df["pincode"] = df["address"].apply(lambda x : x.strip().split("\r\n")[-1].split(" ")[-1])
df["appartment"] = df["address"].apply(lambda x : re.sub(r'\d+', '', x.split("\r\n")[0]))
```

In [ ]:
```
df.city.value_counts()
```

Out[ ]:
```
city
DPO                14289
APO                14060
FPO                14035
East Michael         311
Port Michael         305
                   ...
Port Howardfurt        1
Port Juanshire         1
East Rayburgh          1
Briggsbury             1
Serranoton             1
Name: count, Length: 67513, dtype: int64
```

In [ ]:
```
print('''
Since the distinct count for City is 67600.
It doesn't seems that we'll get learning out from City as of now,
so we will be dicarding them later in data precessing"
''')
```

```
Since the distinct count for City is 67600.
It doesn't seems that we'll get learning out from City as of now,
so we will be dicarding them later in data precessing"
```

In [ ]:
```
state_counts = df.state.value_counts()

# # Create a mask for categories with count less than 5
# mask = df['state'].isin(state_counts[state_counts < 20].index)

# # Assign 'unique' to categories with count less than 5
# df.loc[mask, 'state'] = 'unique'

# df.state.value_counts()
```

In [ ]:
```
df["address"][df["state"]=="Brittanyhaven"]
```

Out[ ]:
```
Series([], Name: address, dtype: object)
```

In [ ]:
```
df.pincode.value_counts()
```

Out[ ]:
```
pincode
70466    56985
30723    56546
22690    56527
48052    55917
00813    45824
29597    45471
05113    45402
11650    11226
93700    11151
86630    10981
Name: count, dtype: int64
```

In [ ]:
```
print(df.appartment.value_counts())
print('''
Since the distinct count for appartment is 2251142.
It doesn't seems that we'll get learning out from appartments as of now,
so we will be dicarding them later in data precessing"
''')
```

```
appartment
Unit  Box                14289
PSC , Box                14060
USCGC Smith                 83
USS Smith                   79
USNS Smith                  68
                         ...
 Oneal Shoals                1
 Arthur Springs Apt.         1
 Benjamin Greens Apt.        1
 Villarreal Ridge            1
 Whitaker Road Suite         1
Name: count, Length: 225142, dtype: int64

Since the distinct count for appartment is 2251142.
It doesn't seems that we'll get learning out from appartments as of now,
so we will be dicarding them later in data precessing"
```

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

Out[ ]:

	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	earlie
0	10000.0	36	11.44	329.48	B	B4	Marketing	10.0	RENT	117000.0	Not Verified	Jan-2015	Fully Paid	vacation	Vacation	26.24	
1	8000.0	36	11.99	265.68	B	B5	Credit analyst	4.0	MORTGAGE	65000.0	Not Verified	Jan-2015	Fully Paid	debt_consolidation	Debt consolidation	22.05	
2	15600.0	36	10.49	506.97	B	B3	Statistician	1.0	RENT	43057.0	Source Verified	Jan-2015	Fully Paid	credit_card	Credit card refinancing	12.79	
3	7200.0	36	6.49	220.65	A	A2	Client Advocate	6.0	RENT	54000.0	Not Verified	Nov-2014	Fully Paid	credit_card	Credit card refinancing	2.60	
4	24375.0	60	17.27	609.33	C	C5	Destiny Management Inc.	9.0	MORTGAGE	55000.0	Verified	Apr-2013	Charged Off	credit_card	Credit Card Refinance	33.95	

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 36 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   loan_amnt             396030 non-null  float64
 1   term                  396030 non-null  int64  
 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  float64
 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 
 27  emp_duration_type     396030 non-null  object 
 28  issue_month           396030 non-null  object 
 29  issue_year            396030 non-null  object 
 30  earliest_cr_month     396030 non-null  object 
 31  earliest_cr_year      396030 non-null  object 
 32  city                  396030 non-null  object 
 33  state                 396030 non-null  object 
 34  pincode               396030 non-null  object 
 35  appartment            396030 non-null  object 
dtypes: float64(13), int64(1), object(22)
memory usage: 108.8+ MB
```

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

Out[ ]:

	loan_amnt	term	int_rate	installment	emp_length	annual_inc	dti	open_acc	pub_rec	revol_bal	revol_util	total_acc	mort_a
count	396030.000000	396030.000000	396030.000000	396030.000000	377729.000000	3.960300e+05	396030.000000	396030.000000	396030.000000	3.960300e+05	395754.000000	396030.000000	358235.0000
mean	14113.888089	41.698053	13.639400	431.849698	6.022566	7.420318e+04	17.379514	11.311153	0.178191	1.584454e+04	53.791749	25.414744	1.8139
std	8357.441341	10.212038	4.472157	250.727790	3.517094	6.163762e+04	18.019092	5.137649	0.530671	2.059184e+04	24.452193	11.886991	2.1479
min	500.000000	36.000000	5.320000	16.080000	1.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000e+00	0.000000	2.000000	0.0000
25%	8000.000000	36.000000	10.490000	250.330000	3.000000	4.500000e+04	11.280000	8.000000	0.000000	6.025000e+03	35.800000	17.000000	0.0000
50%	12000.000000	36.000000	13.330000	375.430000	6.000000	6.400000e+04	16.910000	10.000000	0.000000	1.118100e+04	54.800000	24.000000	1.0000
75%	20000.000000	36.000000	16.490000	567.300000	10.000000	9.000000e+04	22.980000	14.000000	0.000000	1.962000e+04	72.900000	32.000000	3.0000
max	40000.000000	60.000000	30.990000	1533.810000	10.000000	8.706582e+06	9999.000000	90.000000	86.000000	1.743266e+06	892.300000	151.000000	34.0000

In [ ]:
```
df.describe(include="object")
```

Out[ ]:

	grade	sub_grade	emp_title	home_ownership	verification_status	issue_d	loan_status	purpose	title	earliest_cr_line	initial_list_status	application_type	address	emp_
count	396030	396030	373103	396030	396030	396030	396030	396030	394274	396030	396030	396030	396030	
unique	7	35	173105	6	3	115	2	14	48816	684	2	3	393700	
top	B	B3	Teacher	MORTGAGE	Verified	Oct-2014	Fully Paid	debt_consolidation	Debt consolidation	Oct-2000	f	INDIVIDUAL	USS Johnson\r\nFPO AE 48052	
freq	116018	26655	4389	198348	139563	14846	318357	234507	152472	3017	238066	395319	8	

In [ ]:
```
df[df['loan_status']=="Fully Paid"]["emp_title"].value_counts()
```

Out[ ]:
```
emp_title
Teacher                          3532
Manager                          3321
Registered Nurse                 1476
RN                               1467
Supervisor                       1425
                                 ...
Becton and Dickenson                1
EPI USE Labs                        1
Capital Sourcing Leader             1
lupient collision                   1
Seattle United Football Club        1
Name: count, Length: 145235, dtype: int64
```

In [ ]:
```
print(
'''
Object Columns to drop as per my bussiness understanding -
emp_title, issue_d, title, earliest_cr_line, address, city, appartment
'''
)
df.drop(columns=["emp_title", "issue_d", "title", "earliest_cr_line", 'address', 'city', 'appartment'], errors="ignore", inplace=True)
```

```
Object Columns to drop as per my bussiness understanding -
emp_title, issue_d, title, earliest_cr_line, address, city, appartment
```

In [ ]:
```
df.describe(include="object")
```

Out[ ]:

	grade	sub_grade	home_ownership	verification_status	loan_status	purpose	initial_list_status	application_type	emp_duration_type	issue_month	issue_year	earliest_cr_month	earliest
count	396030	396030	396030	396030	396030	396030	396030	396030	396030	396030	396030	396030	
unique	7	35	6	3	2	14	2	3	3	12	10	12	
top	B	B3	MORTGAGE	Verified	Fully Paid	debt_consolidation	f	INDIVIDUAL	equalsto	Oct	2014	Oct	
freq	116018	26655	198348	139563	318357	234507	238066	395319	238264	42130	102860	38291	

In [ ]:
```
df.grade.unique()
```

Out[ ]:
```
array(['B', 'A', 'C', 'E', 'D', 'F', 'G'], dtype=object)
```

In [ ]:
```
df.sub_grade.unique()
```

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

In [ ]:
```
print(df.groupby(["sub_grade"])["loan_status"].value_counts())
```

```
sub_grade  loan_status
A1         Fully Paid     9450
           Charged Off     279
A2         Fully Paid     9106
           Charged Off     461
A3         Fully Paid     9962
                          ...
G3         Fully Paid      270
G4         Fully Paid      206
           Charged Off     168
G5         Charged Off     159
           Fully Paid      157
Name: count, Length: 70, dtype: int64
```

In [ ]:
```
print('''
After looking at the ration it seems A1 is less risky and G5 is more riskier customer,
we'll be doing Label encoding according to that on Grade and Sub-Grade
''')
```

```
After looking at the ration it seems A1 is less risky and G5 is more riskier customer,
we'll be doing Label encoding according to that on Grade and Sub-Grade
```

In [ ]:
```
def encodegrade(value):
    encode_map={
        "A":1,
        "B":2,
        "C":3,
        "D":4,
        "E":5,
        "F":6,
        "G":7
    }
    return encode_map[value]

df["grade"] = df["grade"].apply(encodegrade)
df["sub_grade"] = pd.to_numeric(df["sub_grade"].apply(lambda x: list(x)[-1]))
```

In [ ]:
```
df.describe(include="object")
```

Out[ ]:

	home_ownership	verification_status	loan_status	purpose	initial_list_status	application_type	emp_duration_type	issue_month	issue_year	earliest_cr_month	earliest_cr_year	state	pi
count	396030	396030	396030	396030	396030	396030	396030	396030	396030	396030	396030	396030	3
unique	6	3	2	14	2	3	3	12	10	12	65	54	
top	MORTGAGE	Verified	Fully Paid	debt_consolidation	f	INDIVIDUAL	equalsto	Oct	2014	Oct	2000	AP	
freq	198348	139563	318357	234507	238066	395319	238264	42130	102860	38291	29366	14308	

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

Out[ ]:

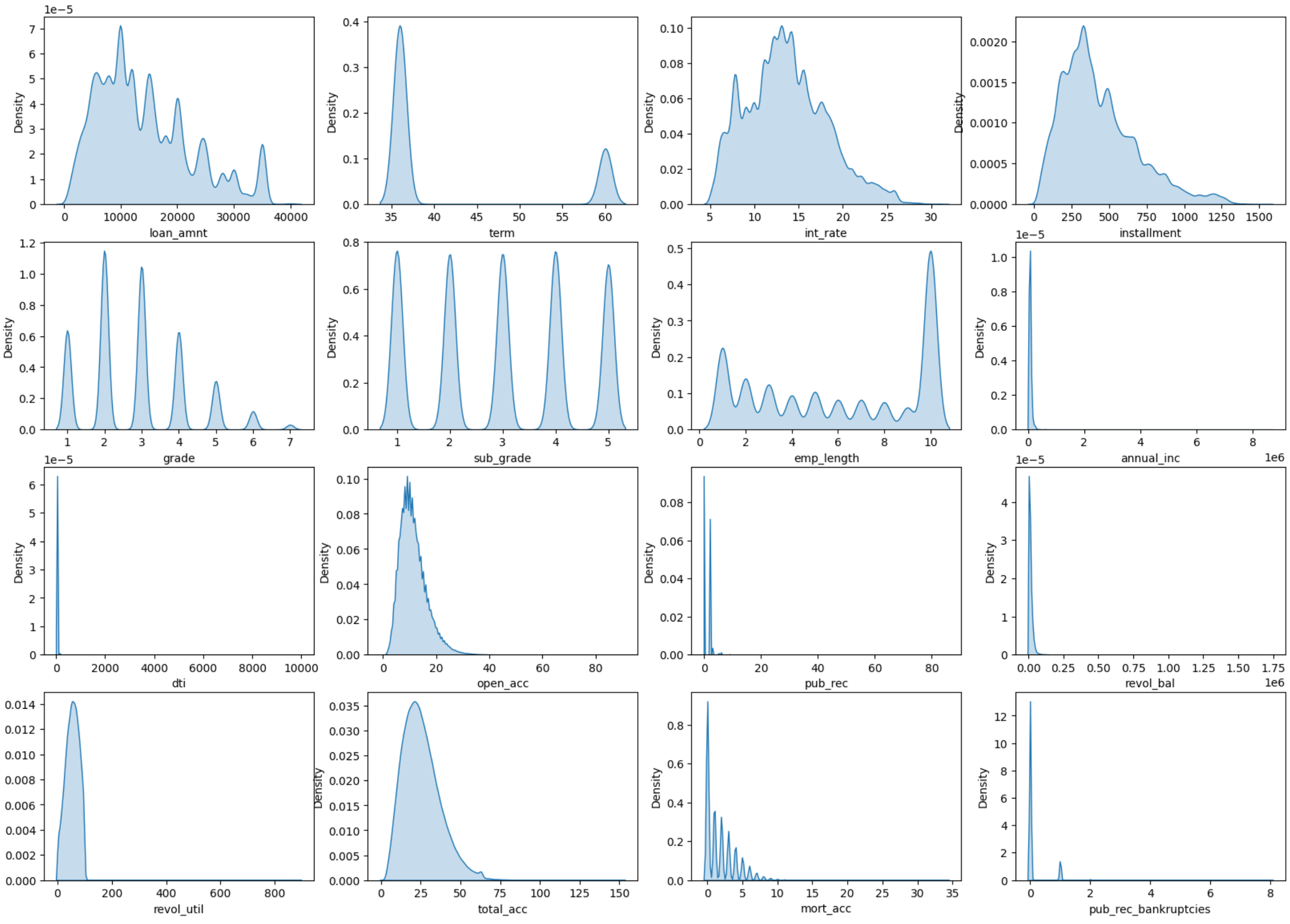
	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_length	annual_inc	dti	open_acc	pub_rec	revol_bal	revol_ut
count	396030.000000	396030.000000	396030.000000	396030.000000	396030.000000	396030.000000	377729.000000	3.960300e+05	396030.000000	396030.000000	396030.000000	3.960300e+05	395754.0000
mean	14113.888089	41.698053	13.639400	431.849698	2.822337	2.971798	6.022566	7.420318e+04	17.379514	11.311153	0.178191	1.584454e+04	53.79174
std	8357.441341	10.212038	4.472157	250.727790	1.333809	1.406773	3.517094	6.163762e+04	18.019092	5.137649	0.530671	2.059184e+04	24.45219
min	500.000000	36.000000	5.320000	16.080000	1.000000	1.000000	1.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000e+00	0.0000
25%	8000.000000	36.000000	10.490000	250.330000	2.000000	2.000000	3.000000	4.500000e+04	11.280000	8.000000	0.000000	6.025000e+03	35.8000
50%	12000.000000	36.000000	13.330000	375.430000	3.000000	3.000000	6.000000	6.400000e+04	16.910000	10.000000	0.000000	1.118100e+04	54.8000
75%	20000.000000	36.000000	16.490000	567.300000	4.000000	4.000000	10.000000	9.000000e+04	22.980000	14.000000	0.000000	1.962000e+04	72.9000
max	40000.000000	60.000000	30.990000	1533.810000	7.000000	5.000000	10.000000	8.706582e+06	9999.000000	90.000000	86.000000	1.743266e+06	892.3000

Visualization

In [ ]:
```
# UNIVARIATE
len(df.describe().columns), df.describe().columns
```

Out[ ]:
```
(16,
 Index(['loan_amnt', 'term', 'int_rate', 'installment', 'grade', 'sub_grade',
        'emp_length', 'annual_inc', 'dti', 'open_acc', 'pub_rec', 'revol_bal',
        'revol_util', 'total_acc', 'mort_acc', 'pub_rec_bankruptcies'],
       dtype='object'))
```

In [ ]:
```
fig=plt.figure(figsize=(20,14)) # width*height
for ind_number, col_name in enumerate(df.describe().columns):
    if ind_number<16:
        plt.subplot(4,4,ind_number+1)
        sns.kdeplot(df[col_name], fill=True)
```

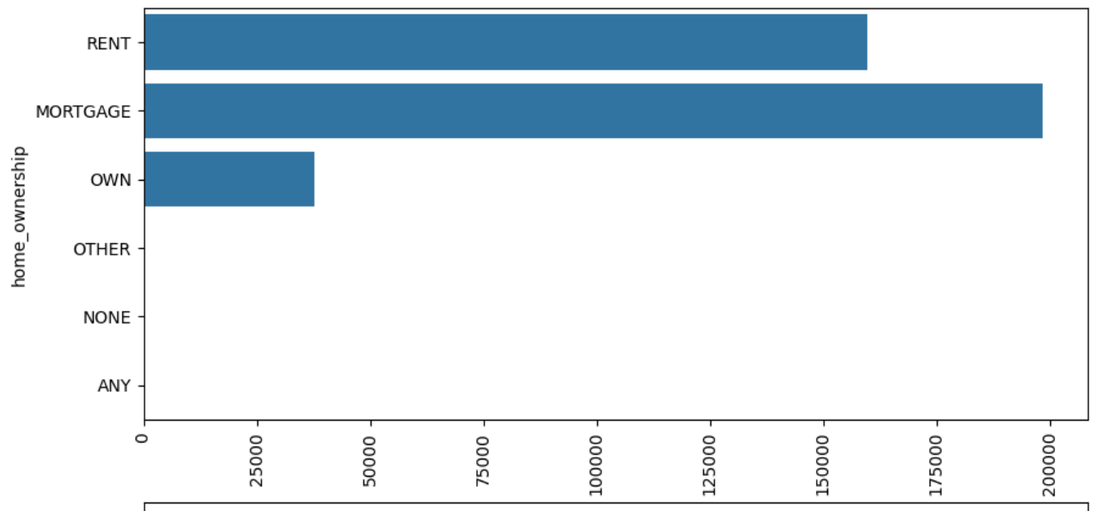


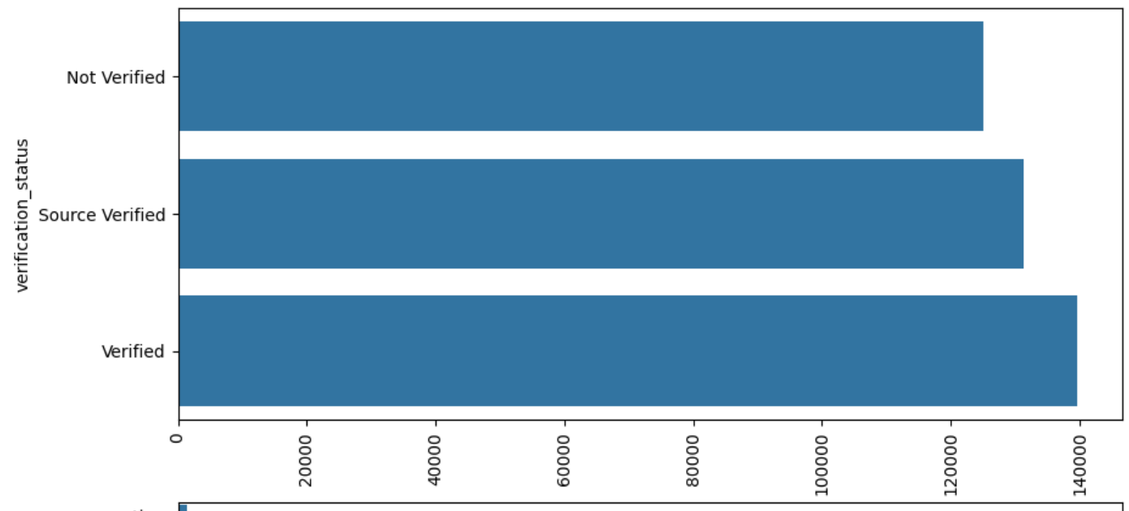
- Distributions like Grade and Sub Grade can be converted to numerical columns based on their impact on the loan status.
- Features like pub_rec, pub_rec_bankruptcies, mort_acc can be converted to categorical columns
- Anual Income, Revol_util, dti, revol_bal have Outliers that can be handled with more care.

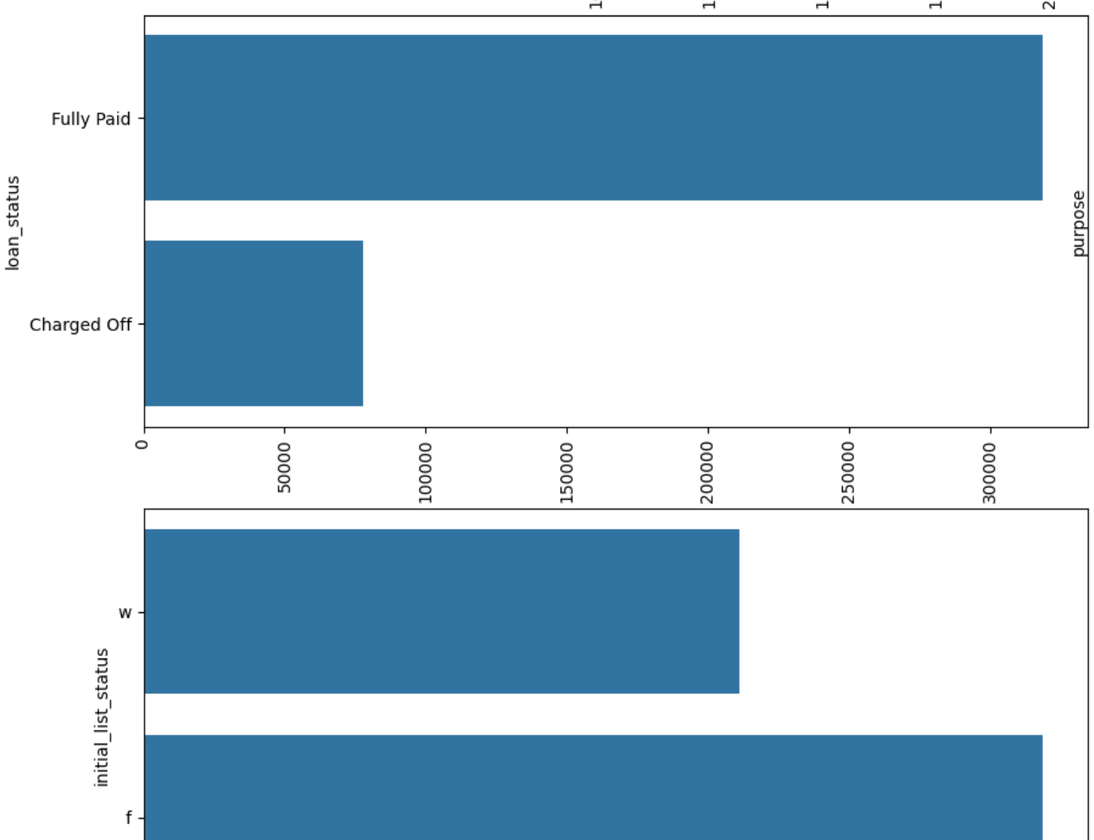
In [ ]:
```
len(df.describe(include="object").columns), df.describe(include="object").columns
```

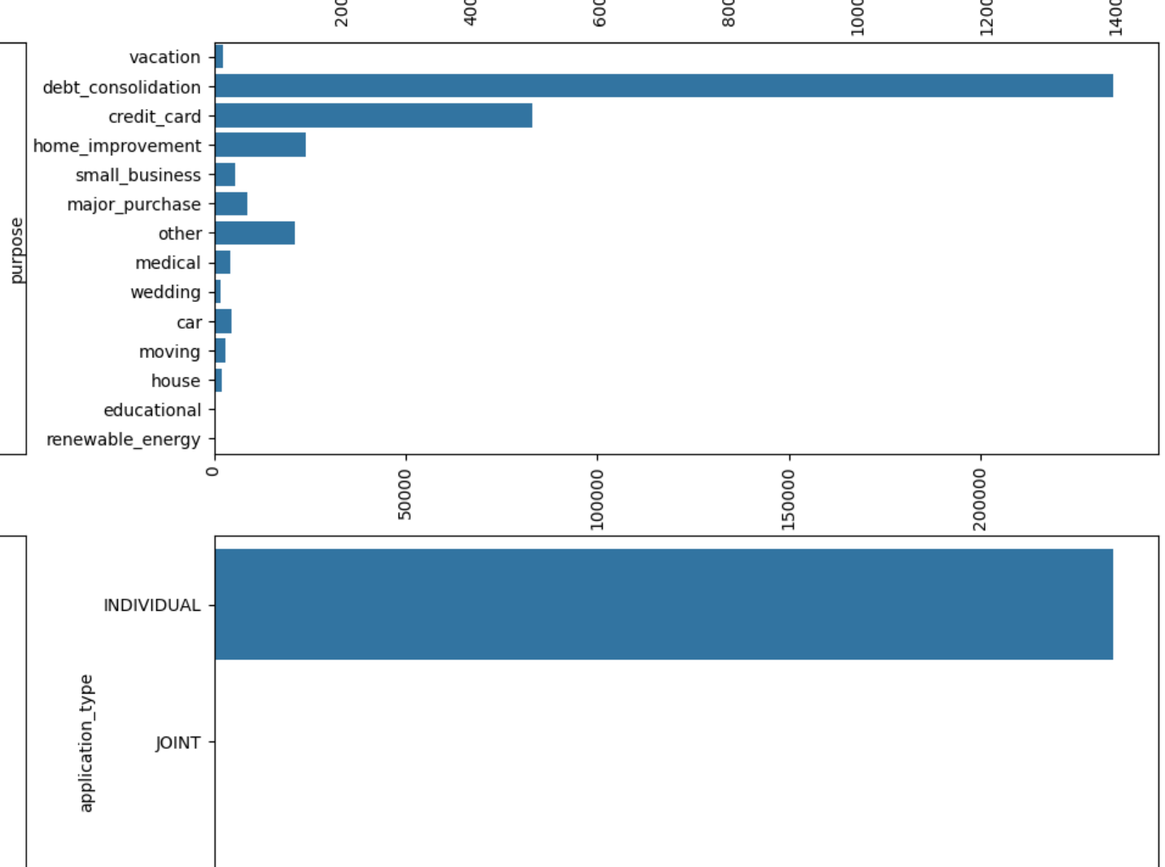
Out[ ]:
```
(13,
 Index(['home_ownership', 'verification_status', 'loan_status', 'purpose',
        'initial_list_status', 'application_type', 'emp_duration_type',
        'issue_month', 'issue_year', 'earliest_cr_month', 'earliest_cr_year',
        'state', 'pincode'],
       dtype='object'))
```

In [ ]:
```
fig=plt.figure(figsize=(22,36)) # width*height
for ind_number, col_name in enumerate(df.describe(include="object").columns):
    if ind_number<13:
        plt.subplot(7,2,ind_number+1)
        sns.countplot(df[col_name], fill=True)
        plt.xticks(rotation=90)
```


home_ownership
RENT
MORTGAGE
OWN
OTHER
NONE
ANY
0
25000
50000
75000
100000
125000
150000
175000
200000

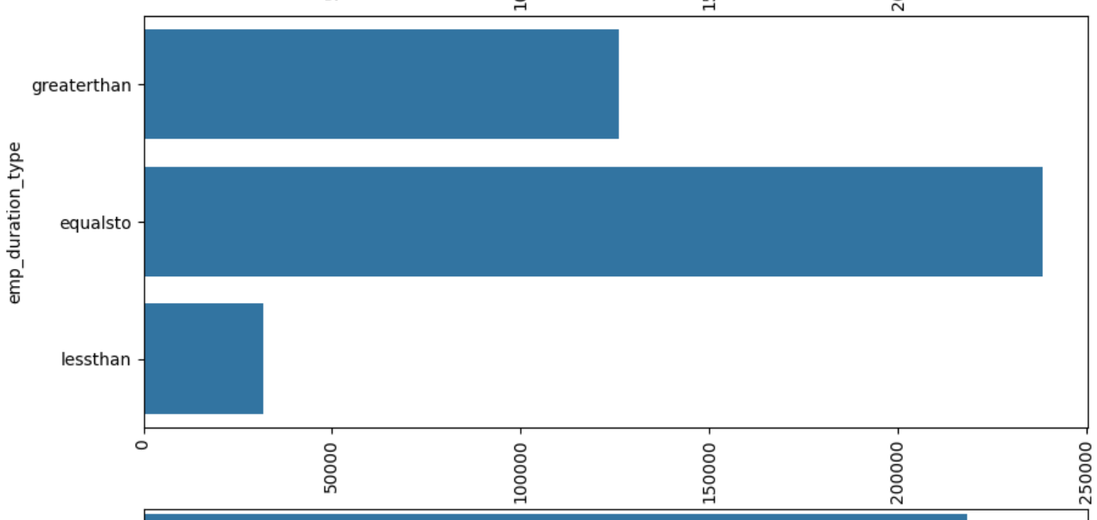

verification_status
Not Verified
Source Verified
Verified
0
20000
40000
60000
80000
100000
120000
140000

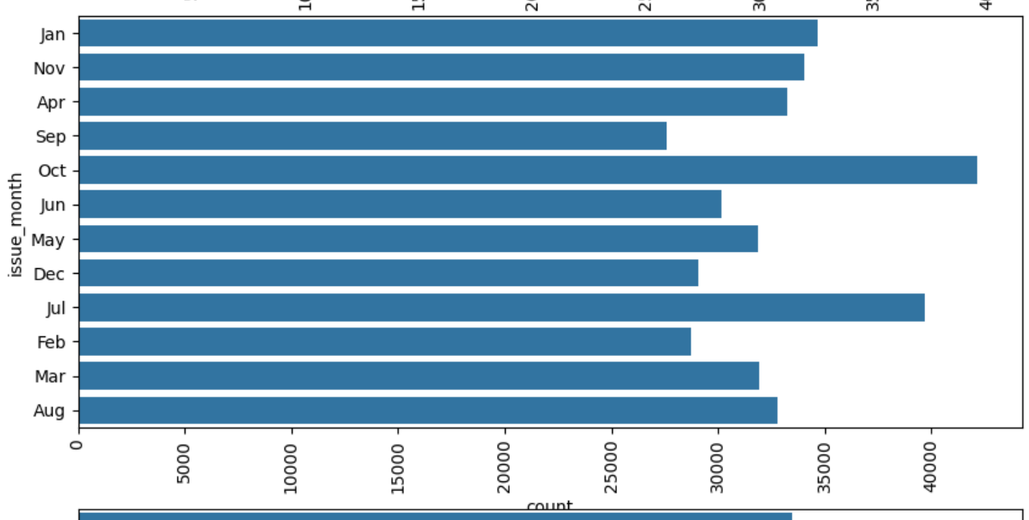

loan_status
Fully Paid
Charged Off
0
50000
100000
150000
200000
250000
300000

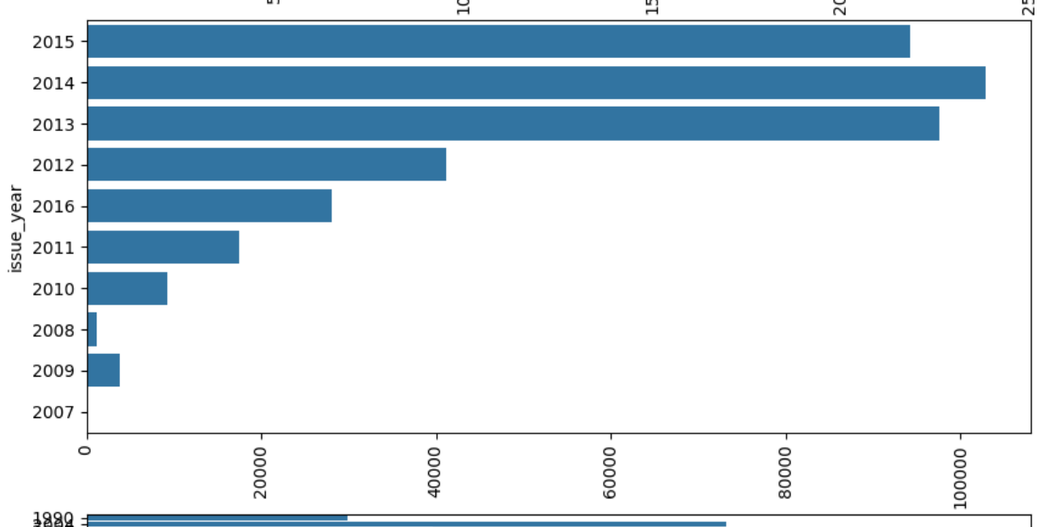

purpose
vacation
debt_consolidation
credit_card
home_improvement
small_business
major_purchase
other
medical
wedding
car
moving
house
educational
renewable_energy
0
50000
100000
150000
200000

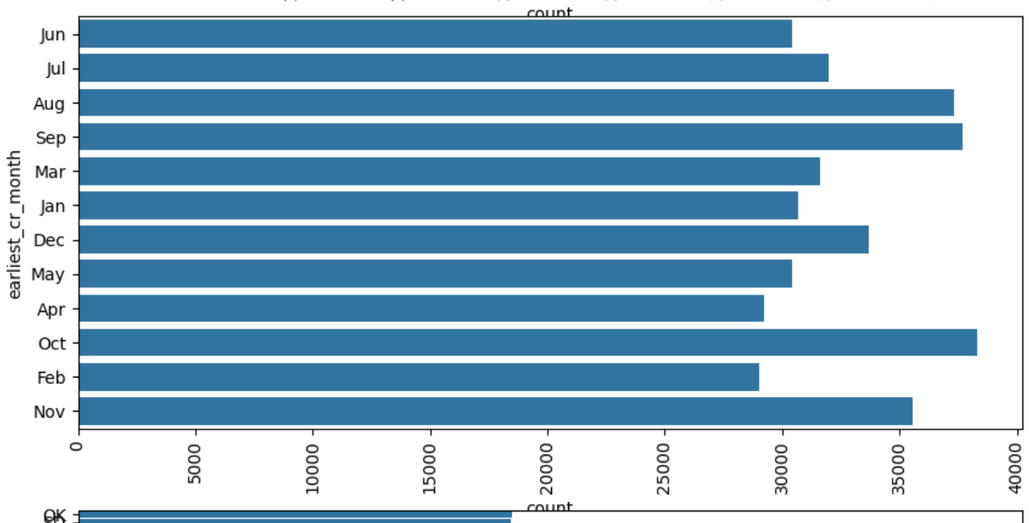
initial_list_status
w
f
0
50000
100000
150000
200000

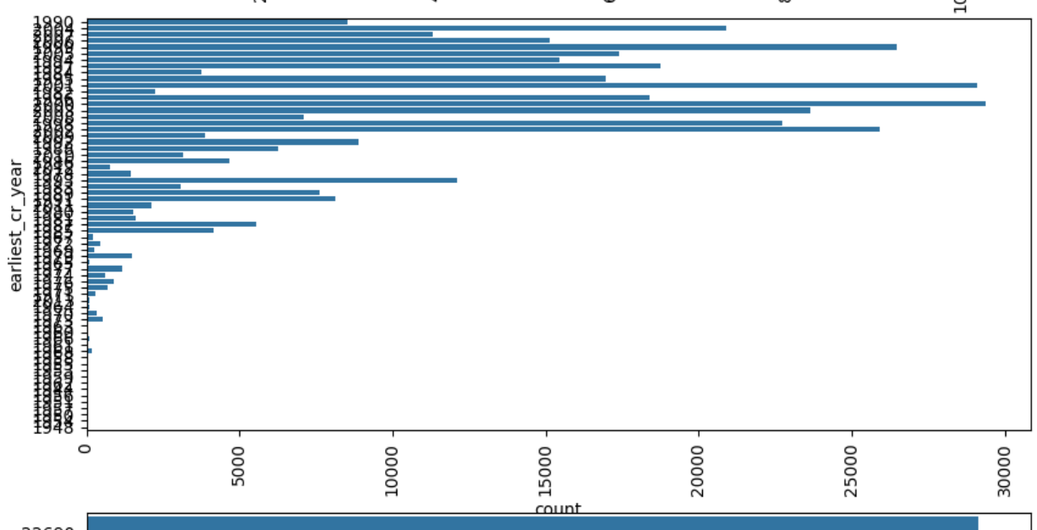
application_type
INDIVIDUAL
JOINT
DIRECT_PAY
0
50000
100000
150000
200000
250000
300000
350000
400000

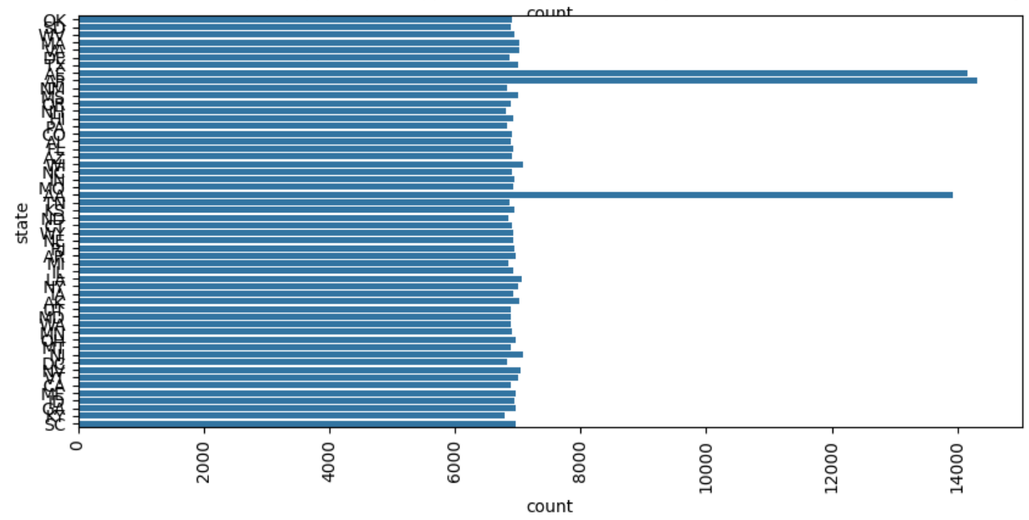

emp_duration_type
greaterthan
equalsto
lessthan
0
50000
100000
150000
200000
250000

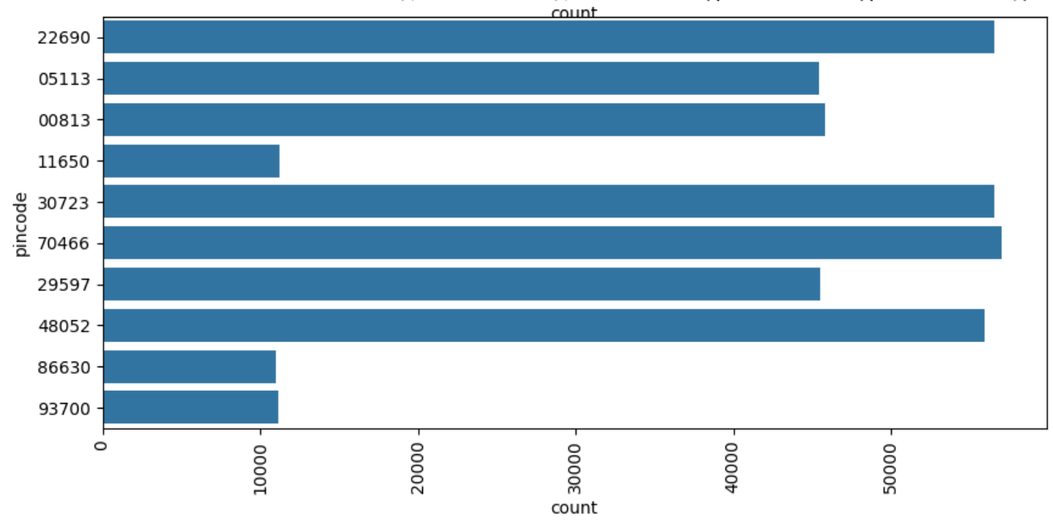

issue_month
Jan
Nov
Apr
Sep
Oct
Jun
May
Dec
Jul
Feb
Mar
Aug
0
5000
10000
15000
20000
25000
30000
35000
40000
count


issue_year
2015
2014
2013
2012
2016
2011
2010
2008
2009
2007
0
20000
40000
60000
80000
100000


earliest_cr_month
Jun
Jul
Aug
Sep
Mar
Jan
Dec
May
Apr
Oct
Feb
Nov
0
5000
10000
15000
20000
25000
30000
35000
40000
count


earliest_cr_year
0
5000
10000
15000
20000
25000
30000
count


state
0
2000
4000
6000
8000
10000
12000
14000
count

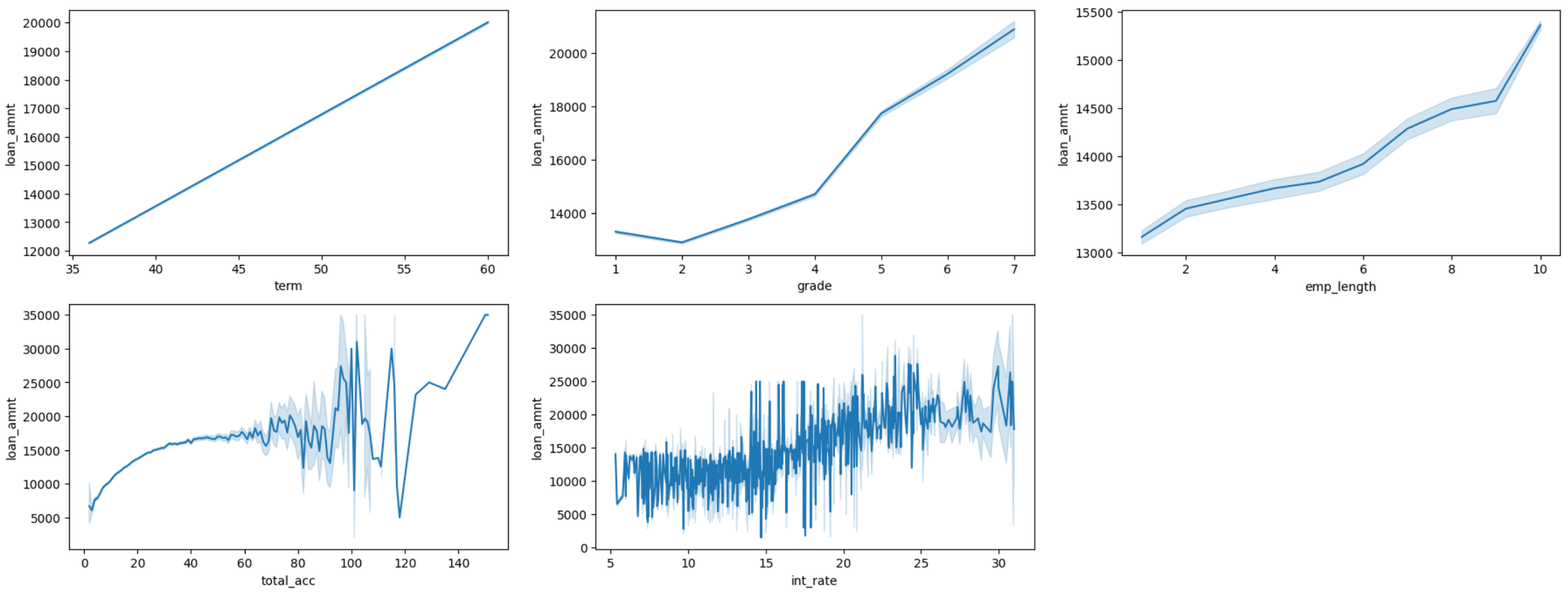

pincode
22690
05113
00813
11650
30723
70466
29597
48052
86630
93700
0
10000
20000
30000
40000
50000
count

In [ ]:
```
print(
'''
BIVARIATE Analysis:

1. CONTINUOUS VS CONTINUOUS
2. CONTINUOUS VS CATEGORICAL
3. CATEGORICAL VS CATEGORICAL
''')
```

```
BIVARIATE Analysis:

1. CONTINUOUS VS CONTINUOUS
2. CONTINUOUS VS CATEGORICAL
3. CATEGORICAL VS CATEGORICAL
```

In [ ]:
```
# 1. CONTINUOUS VS CONTINUOUS

fig=plt.figure(figsize=(22,8))

plt.subplot(2,3,1)
sns.lineplot(y='loan_amnt', x='term', data=df)

plt.subplot(2,3,2)
sns.lineplot(y='loan_amnt', x='grade', data=df)

plt.subplot(2,3,3)
sns.lineplot(y='loan_amnt', x='emp_length', data=df)

plt.subplot(2,3,4)
sns.lineplot(y='loan_amnt', x='total_acc', data=df)

plt.subplot(2,3,5)
sns.lineplot(y='loan_amnt', x='int_rate', data=df)
```

Out[ ]: <Axes: xlabel='int_rate', ylabel='loan_amnt'>



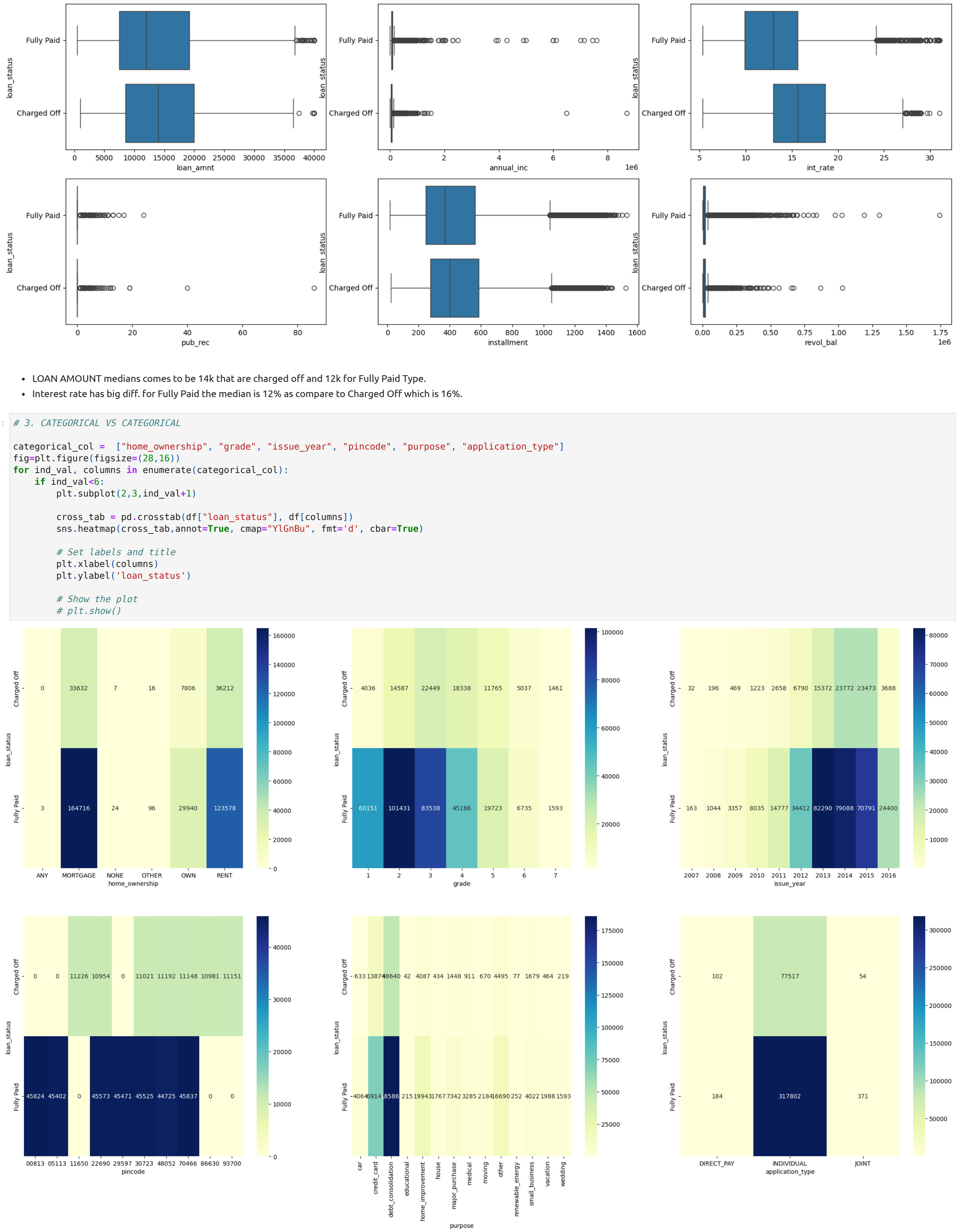
- Positive correlation between Term and Loan Amount
- Total credit account has positive correlation with Loan amount, stating that more the credit account more will be the Loan Amount.
- Interest rate increases with increase in Loan Amount

In [ ]:
```
# Calculate Pearson correlation
pearson_corr = df['loan_amnt'].corr(df['installment'])
print(f"Pearson Correlation: {pearson_corr}")

# Calculate Spearman correlation
spearman_corr = df['loan_amnt'].corr(df['installment'], method='spearman')
print(f"Spearman Correlation: {spearman_corr}")
```

```
Pearson Correlation: 0.953928908261621
Spearman Correlation: 0.9683337077962264
```

In [ ]:
```
# 2. CONTINUOUS VS CATEGORICAL
fig=plt.figure(figsize=(22,8))

plt.subplot(2,3,1)
sns.boxplot(y='loan_status', x='loan_amnt', data=df)

plt.subplot(2,3,2)
sns.boxplot(y='loan_status', x='annual_inc', data=df)

plt.subplot(2,3,3)
sns.boxplot(y='loan_status', x='int_rate', data=df)

plt.subplot(2,3,4)
sns.boxplot(y='loan_status', x='pub_rec', data=df)

plt.subplot(2,3,5)
sns.boxplot(y='loan_status', x='installment', data=df)

plt.subplot(2,3,6)
sns.boxplot(y='loan_status', x='revol_bal', data=df)
```

Out[ ]: <Axes: xlabel='revol_bal', ylabel='loan_status'>



- LOAN AMOUNT medians comes to be 14k that are charged off and 12k for Fully Paid Type.
- Interest rate has big diff. for Fully Paid the median is 12% as compare to Charged Off which is 16%.

In [ ]:
```
# 3. CATEGORICAL VS CATEGORICAL

categorical_col =  ["home_ownership", "grade", "issue_year", "pincode", "purpose", "application_type"]
fig=plt.figure(figsize=(28,16))
for ind_val, columns in enumerate(categorical_col):
    if ind_val<6:
        plt.subplot(2,3,ind_val+1)

        cross_tab = pd.crosstab(df["loan_status"], df[columns])
        sns.heatmap(cross_tab,annot=True, cmap="YlGnBu", fmt='d', cbar=True)

        # Set labels and title
        plt.xlabel(columns)
        plt.ylabel('loan_status')

        # Show the plot
        # plt.show()
```


- Mortages have high chances of paying the loan
- INDIVIDUAL type application have more chances of buying loan as compare to any other segment.
- DEBT_CONSOLIDATION is main reason for taking loan

In [ ]:

Data Preprocessing

Duplicate and Missing value treatment

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

Out[ ]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_length	home_ownership	annual_inc	verification_status	loan_status	purpose	dti	open_acc	pub_rec	revol_bal	revol_util	t
0	10000.0	36	11.44	329.48	2	4	10.0	RENT	117000.0	Not Verified	Fully Paid	vacation	26.24	16.0	0.0	36369.0	41.8	
1	8000.0	36	11.99	265.68	2	5	4.0	MORTGAGE	65000.0	Not Verified	Fully Paid	debt_consolidation	22.05	17.0	0.0	20131.0	53.3	
2	15600.0	36	10.49	506.97	2	3	1.0	RENT	43057.0	Source Verified	Fully Paid	credit_card	12.79	13.0	0.0	11987.0	92.2	
3	7200.0	36	6.49	220.65	1	2	6.0	RENT	54000.0	Not Verified	Fully Paid	credit_card	2.60	6.0	0.0	5472.0	21.5	
4	24375.0	60	17.27	609.33	3	5	9.0	MORTGAGE	55000.0	Verified	Charged Off	credit_card	33.95	13.0	0.0	24584.0	69.8	

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

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

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

Out[ ]:
```
0
```

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

Out[ ]:
```
loan_amnt                 0
term                      0
int_rate                  0
installment               0
grade                     0
sub_grade                 0
emp_length            18301
home_ownership            0
annual_inc                0
verification_status       0
loan_status               0
purpose                   0
dti                       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
emp_duration_type         0
issue_month               0
issue_year                0
earliest_cr_month         0
earliest_cr_year          0
state                     0
pincode                   0
dtype: int64
```

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

Out[ ]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_length	annual_inc	dti	open_acc	pub_rec	revol_bal	revol_ul
count	396030.000000	396030.000000	396030.000000	396030.000000	396030.000000	396030.000000	377729.000000	3.960300e+05	396030.000000	396030.000000	396030.000000	3.960300e+05	395754.0000(
mean	14113.888089	41.698053	13.639400	431.849698	2.822337	2.971798	6.022566	7.420318e+04	17.379514	11.311153	0.178191	1.584454e+04	53.79174
std	8357.441341	10.212038	4.472157	250.727790	1.333809	1.406773	3.517094	6.163762e+04	18.019092	5.137649	0.530671	2.059184e+04	24.45219
min	500.000000	36.000000	5.320000	16.080000	1.000000	1.000000	1.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000e+00	0.0000(
25%	8000.000000	36.000000	10.490000	250.330000	2.000000	2.000000	3.000000	4.500000e+04	11.280000	8.000000	0.000000	6.025000e+03	35.8000(
50%	12000.000000	36.000000	13.330000	375.430000	3.000000	3.000000	6.000000	6.400000e+04	16.910000	10.000000	0.000000	1.118100e+04	54.8000(
75%	20000.000000	36.000000	16.490000	567.300000	4.000000	4.000000	10.000000	9.000000e+04	22.980000	14.000000	0.000000	1.962000e+04	72.9000(
max	40000.000000	60.000000	30.990000	1533.810000	7.000000	5.000000	10.000000	8.706582e+06	9999.000000	90.000000	86.000000	1.743266e+06	892.3000(

In [ ]:
```
df.describe(include="object")
```

Out[ ]:

	home_ownership	verification_status	loan_status	purpose	initial_list_status	application_type	emp_duration_type	issue_month	issue_year	earliest_cr_month	earliest_cr_year	state	pi
count	396030	396030	396030	396030	396030	396030	396030	396030	396030	396030	396030	396030	3
unique	6	3	2	14	2	3	3	12	10	12	65	54	
top	MORTGAGE	Verified	Fully Paid	debt_consolidation	f	INDIVIDUAL	equalsto	Oct	2014	Oct	2000	AP	
freq	198348	139563	318357	234507	238066	395319	238264	42130	102860	38291	29366	14308	

In [ ]:
```
from sklearn.impute import KNNImputer

# Handling missing emp_length
# Identify numeric and categorical columns
# Instead of selecting all numeric columns, seting as per business understanding
# numeric_columns = df.select_dtypes(include='number').columns
numeric_columns = ["loan_amnt", "annual_inc", "emp_length", "mort_acc"]

# Extract the numeric part of the DataFrame
df_numeric = df[numeric_columns]

# Impute missing values in the numeric part using KNNImputer
knn_imputer = KNNImputer(n_neighbors=10)
df_numeric_imputed = pd.DataFrame(knn_imputer.fit_transform(df_numeric), columns=numeric_columns)

# Combine the imputed numeric part with the original categorical part
df = pd.concat([df.drop(columns=numeric_columns), df_numeric_imputed], axis=1)

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

Out[ ]:
```
term                    0
int_rate                0
installment             0
grade                   0
sub_grade               0
home_ownership          0
verification_status     0
loan_status             0
purpose                 0
dti                     0
open_acc                0
pub_rec                 0
revol_bal               0
revol_util            276
total_acc               0
initial_list_status     0
application_type        0
pub_rec_bankruptcies  535
emp_duration_type       0
issue_month             0
issue_year              0
earliest_cr_month       0
earliest_cr_year        0
state                   0
pincode                 0
loan_amnt               0
annual_inc              0
emp_length              0
mort_acc                0
dtype: int64
```

In [ ]:
```
# Handling missing revol_util

# Identify numeric and categorical columns
# Instead of selecting all numeric columns, seting as per business understanding
# numeric_columns = df.select_dtypes(include='number').columns
numeric_columns = ["revol_bal", "total_acc", "revol_util"]

# Extract the numeric part of the DataFrame
df_numeric = df[numeric_columns]

# Impute missing values in the numeric part using KNNImputer
knn_imputer = KNNImputer(n_neighbors=10)
df_numeric_imputed = pd.DataFrame(knn_imputer.fit_transform(df_numeric), columns=numeric_columns)

# Combine the imputed numeric part with the original categorical part
df = pd.concat([df.drop(columns=numeric_columns), df_numeric_imputed], axis=1)

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

Out[ ]:
```
term                    0
int_rate                0
installment             0
grade                   0
sub_grade               0
home_ownership          0
verification_status     0
loan_status             0
purpose                 0
dti                     0
open_acc                0
pub_rec                 0
initial_list_status     0
application_type        0
pub_rec_bankruptcies  535
emp_duration_type       0
issue_month             0
issue_year              0
earliest_cr_month       0
earliest_cr_year        0
state                   0
pincode                 0
loan_amnt               0
annual_inc              0
emp_length              0
mort_acc                0
revol_bal               0
total_acc               0
revol_util              0
dtype: int64
```

In [ ]:
```
# Handling missing pub_rec_bankruptcies
## Dropping pub_rec_bankruptcies as does find any correlation with other feature.
df.dropna(inplace=True)
df.isnull().sum()
```

Out[ ]:
```
term                  0
int_rate              0
installment           0
grade                 0
sub_grade             0
home_ownership        0
verification_status   0
loan_status           0
purpose               0
dti                   0
open_acc              0
pub_rec               0
initial_list_status   0
application_type      0
pub_rec_bankruptcies  0
emp_duration_type     0
issue_month           0
issue_year            0
earliest_cr_month     0
earliest_cr_year      0
state                 0
pincode               0
loan_amnt             0
annual_inc            0
emp_length            0
mort_acc              0
revol_bal             0
total_acc             0
revol_util            0
dtype: int64
```

In [ ]:
```
df.to_csv("dataset/df_processed_phase1.csv", index=False)
```

Outlier Treatment

In [ ]:
```
df = pd.read_csv("dataset/df_processed_phase1.csv")
df.head()
```

Out[ ]:

	term	int_rate	installment	grade	sub_grade	home_ownership	verification_status	loan_status	purpose	dti	open_acc	pub_rec	initial_list_status	application_type	pub_rec_bankruptcies	e
0	36	11.44	329.48	2	4	RENT	Not Verified	Fully Paid	vacation	26.24	16.0	0.0	w	INDIVIDUAL	0.0	
1	36	11.99	265.68	2	5	MORTGAGE	Not Verified	Fully Paid	debt_consolidation	22.05	17.0	0.0	f	INDIVIDUAL	0.0	
2	36	10.49	506.97	2	3	RENT	Source Verified	Fully Paid	credit_card	12.79	13.0	0.0	f	INDIVIDUAL	0.0	
3	36	6.49	220.65	1	2	RENT	Not Verified	Fully Paid	credit_card	2.60	6.0	0.0	f	INDIVIDUAL	0.0	
4	60	17.27	609.33	3	5	MORTGAGE	Verified	Charged Off	credit_card	33.95	13.0	0.0	f	INDIVIDUAL	0.0	

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

Out[ ]:
```
(395495, 29)
```

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

Out[ ]:
```
term                    0
int_rate                0
installment             0
grade                   0
sub_grade               0
home_ownership          0
verification_status     0
loan_status             0
purpose                 0
dti                     0
open_acc                0
pub_rec                 0
initial_list_status     0
application_type        0
pub_rec_bankruptcies    0
emp_duration_type       0
issue_month             0
issue_year              0
earliest_cr_month       0
earliest_cr_year        0
state                   0
pincode                 0
loan_amnt               0
annual_inc              0
emp_length              0
mort_acc                0
revol_bal               0
total_acc               0
revol_util              0
dtype: int64
```

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

Out[ ]:
```
Index(['term', 'int_rate', 'installment', 'grade', 'sub_grade', 'dti',
       'open_acc', 'pub_rec', 'pub_rec_bankruptcies', 'issue_year',
       'earliest_cr_year', 'pincode', 'loan_amnt', 'annual_inc', 'emp_length',
       'mort_acc', 'revol_bal', 'total_acc', 'revol_util'],
      dtype='object')
```

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

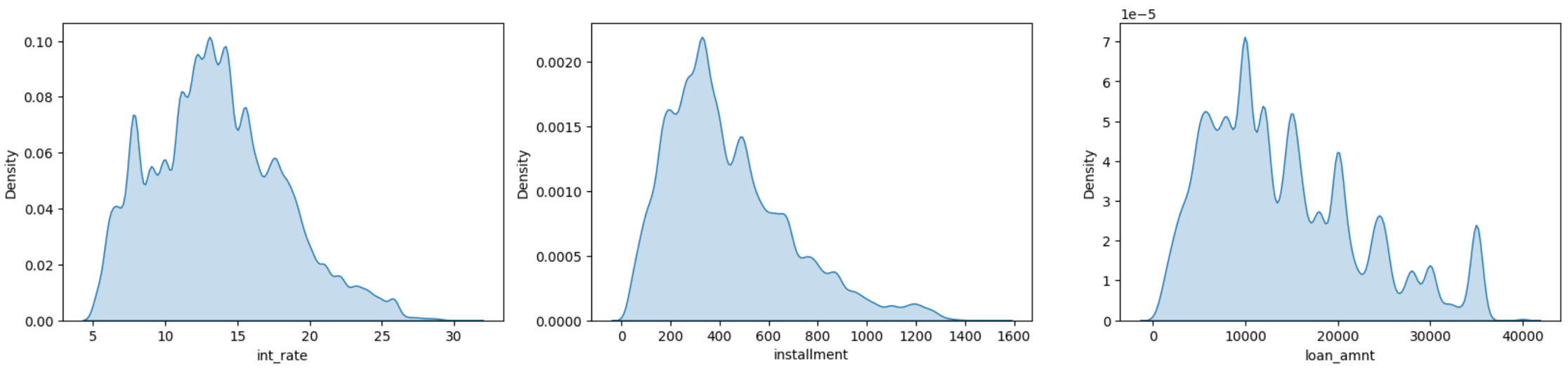
Out[ ]:

	term	int_rate	installment	grade	sub_grade	dti	open_acc	pub_rec	pub_rec_bankruptcies	issue_year	earliest_cr_year	pincode	3
count	395495.000000	395495.000000	395495.000000	395495.000000	395495.000000	395495.000000	395495.000000	395495.000000	395495.000000	395495.000000	395495.000000	395495.000000	
mean	41.705761	13.643530	432.007574	2.822620	2.971926	17.388523	11.313754	0.178384	0.121648	2013.637164	1997.861485	34001.141289	1
std	10.216791	4.473081	250.740725	1.333844	1.406788	18.027900	5.136860	0.530955	0.356174	1.466190	7.198201	25607.536565	
min	36.000000	5.320000	16.080000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	2007.000000	1944.000000	813.000000	
25%	36.000000	10.490000	250.330000	2.000000	2.000000	11.295000	8.000000	0.000000	0.000000	2013.000000	1994.000000	11650.000000	
50%	36.000000	13.330000	375.450000	3.000000	3.000000	16.910000	10.000000	0.000000	0.000000	2014.000000	1999.000000	29597.000000	1
75%	36.000000	16.550000	567.730000	4.000000	4.000000	22.990000	14.000000	0.000000	0.000000	2015.000000	2003.000000	48052.000000	2
max	60.000000	30.990000	1533.810000	7.000000	5.000000	9999.000000	90.000000	86.000000	8.000000	2016.000000	2013.000000	93700.000000	4

In [ ]:
```
numeric_col1 = ["open_acc", "open_acc", "total_acc", "revol_util", "dti"]
numeric_col4 = ["annual_inc", "revol_bal"]
```

In [ ]:
```
# 1. int_rate, installment, loan_amnt doesn't seem to have big variation in values, looks it's okay not to consider them in outliers detection

fig=plt.figure(figsize=(20,4))

plt.subplot(1,3,1)
sns.kdeplot(df["int_rate"], fill=True)

plt.subplot(1,3,2)
sns.kdeplot(df["installment"], fill=True)

plt.subplot(1,3,3)
sns.kdeplot(df["loan_amnt"], fill=True)
```

Out[ ]:
```
<Axes: xlabel='loan_amnt', ylabel='Density'>
```

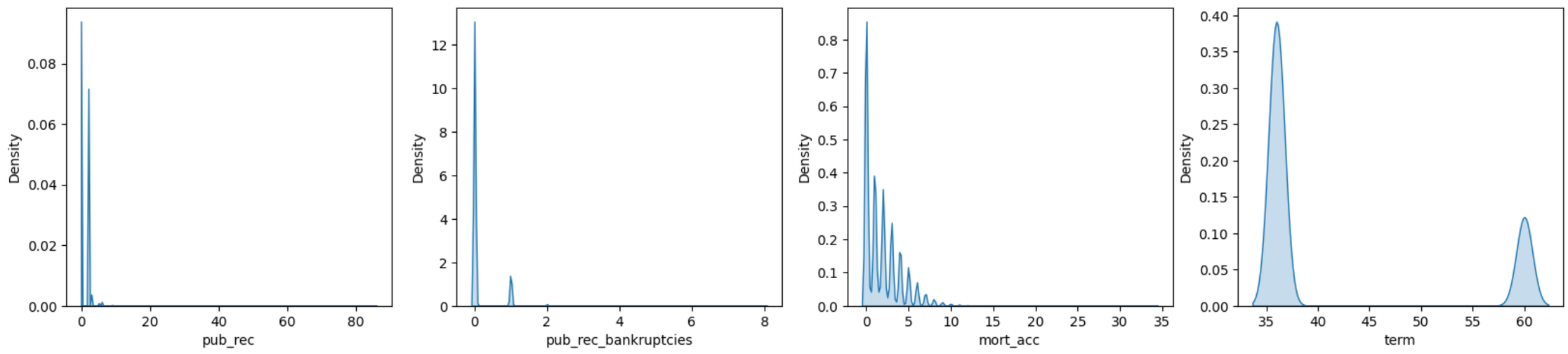


- Interest rate, Installment and Loan Amount seems to be left skewed

In [ ]:
```
# 2. Few Features have outside values but we can not consider them as outliears, it is better to create some flag(binary flag at some condition) for them as they may behave l
# Like pub_rec, pub_rec_bankruptcies, mort_acc.

fig=plt.figure(figsize=(20,4))

plt.subplot(1,4,1)
sns.kdeplot(df["pub_rec"], fill=True)
```

```
plt.subplot(1,4,2)
sns.kdeplot(df["pub_rec_bankruptcies"], fill=True)

plt.subplot(1,4,3)
sns.kdeplot(df["mort_acc"], fill=True)

plt.subplot(1,4,4)
sns.kdeplot(df["term"], fill=True)
```

Out[ ]: <Axes: xlabel='term', ylabel='Density'>



- After analysing it looks like generating binary flag is better for above features.

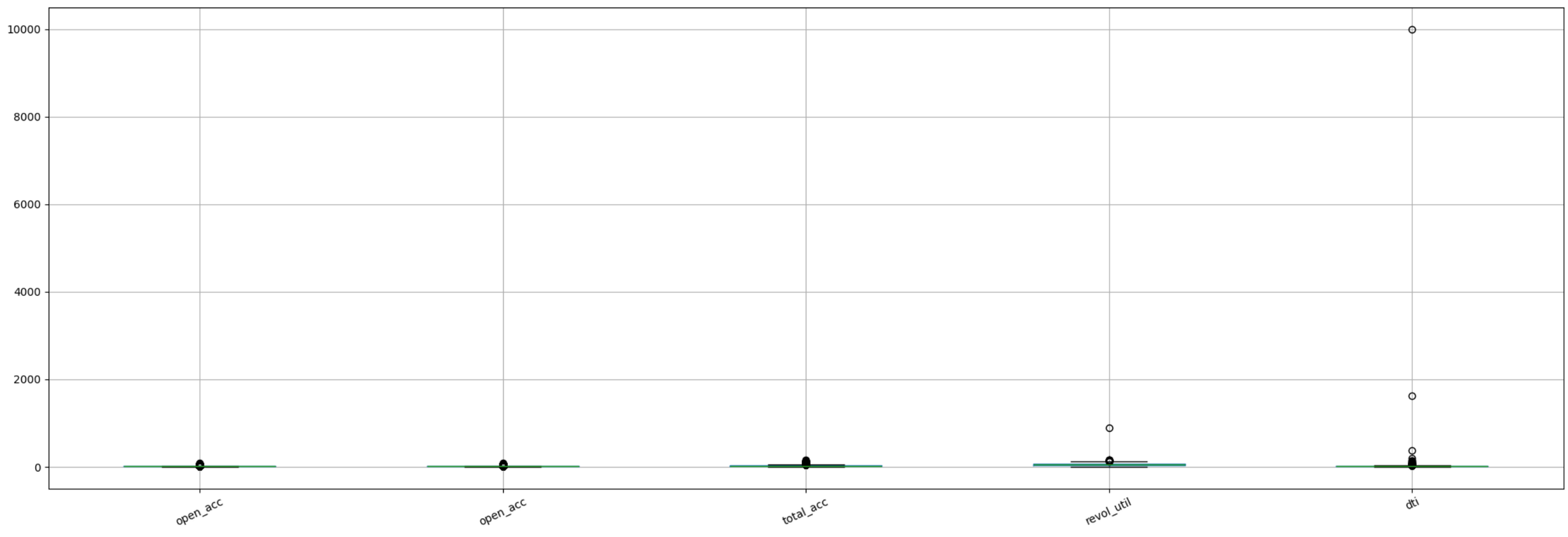
In [ ]:
```
df["pub_rec"]= df["pub_rec"].apply(lambda x: "good" if x==0 else "bad")
df["pub_rec_bankruptcies"]= df["pub_rec_bankruptcies"].apply(lambda x: "good" if x==0 else "bad")
df["mort_acc"]= df["mort_acc"].apply(lambda x: "good" if x==0 else "bad")
df["term"]= df["term"].apply(lambda x: "36" if x==36 else "60")
```

In [ ]:
```
df.describe(include="object")
```

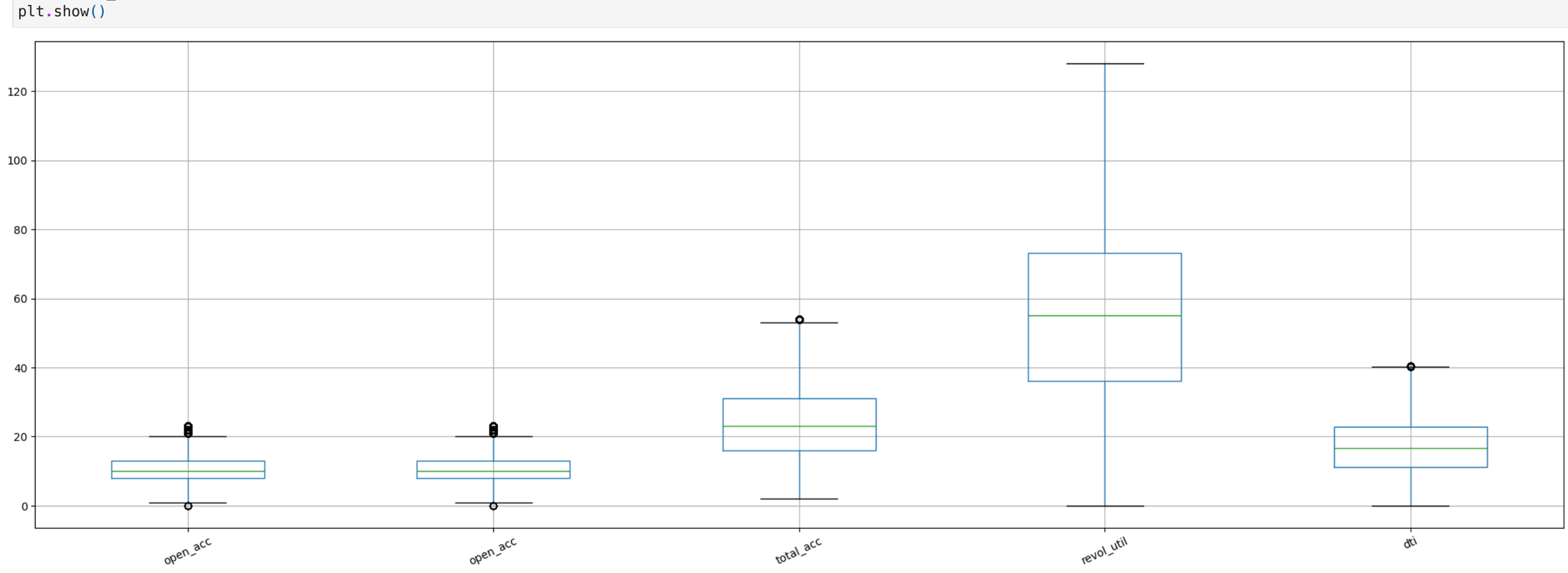
Out[ ]:

	term	home_ownership	verification_status	loan_status	purpose	pub_rec	initial_list_status	application_type	pub_rec_bankruptcies	emp_duration_type	issue_month	earliest_cr_month
count	395495	395495	395495	395495	395495	395495	395495	395495	395495	395495	395495	395495
unique	2	6	3	2	14	2	2	3	2	3	12	12
top	36	MORTGAGE	Verified	Fully Paid	debt_consolidation	good	f	INDIVIDUAL	good	equalsto	Oct	Oct
freq	301470	198151	139562	317909	234292	337755	237531	394784	350380	237949	42097	38248

In [ ]:
```
df[numeric_col1].boxplot(rot=25, figsize=(25,8))
plt.show()
```



In [ ]:
```
Q1 = df[numeric_col1].quantile(0.25)
Q3 = df[numeric_col1].quantile(0.75)
IQR = Q3 - Q1

df = df[~((df[numeric_col1] < (Q1 - 1.5 * IQR)) | (df[numeric_col1] > (Q3 + 1.5 * IQR))).any(axis=1)]
df = df.reset_index(drop=True)

df[numeric_col1].boxplot(rot=25, figsize=(25,8))
plt.show()
```

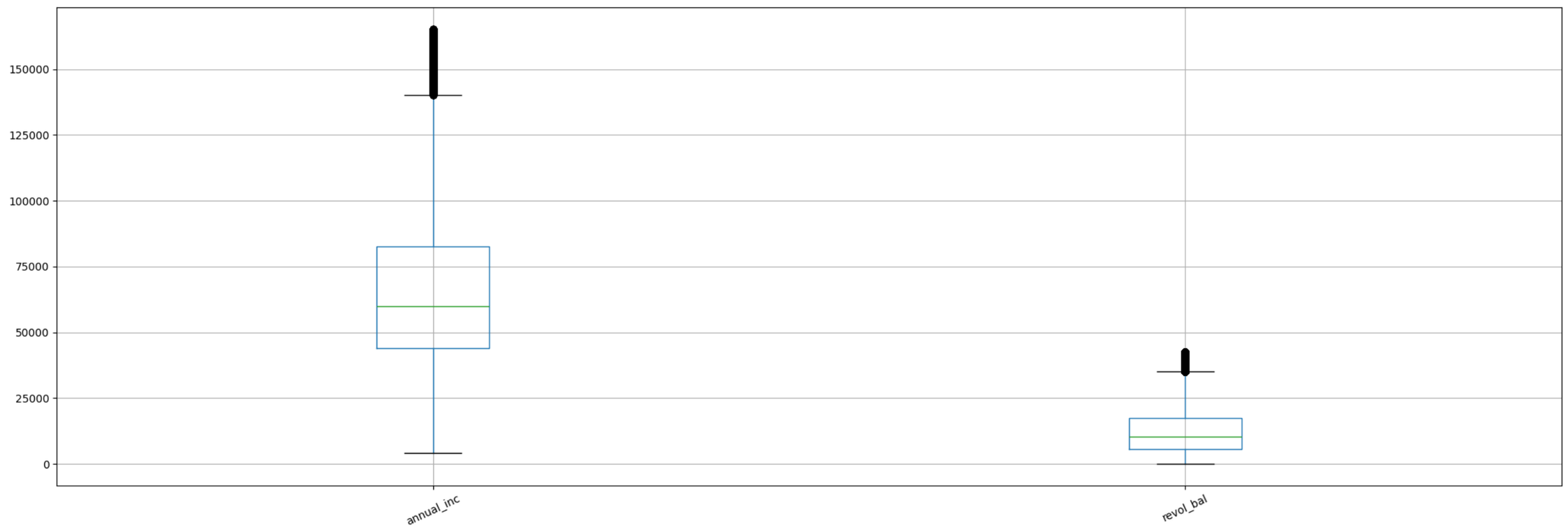


In [ ]:
```
df.shape
```

Out[ ]: (379612, 29)

In [ ]:
```
df[numeric_col4].boxplot(rot=25, figsize=(25,8))
plt.show()
```

In [ ]:
```
Q1 = df[numeric_col4].quantile(0.25)
Q3 = df[numeric_col4].quantile(0.75)
IQR = Q3 - Q1

df = df[~((df[numeric_col4] < (Q1 - 1.75 * IQR)) | (df[numeric_col4] > (Q3 + 1.75 * IQR))).any(axis=1)]
df = df.reset_index(drop=True)

df[numeric_col4].boxplot(rot=25, figsize=(25,8))
plt.show()
```



In [ ]:
```
# Selecting INDIVIDUAL application type, as we are dealing with PERSONNEL LOAN

df = df[df["application_type"] == "INDIVIDUAL"]
df.drop(columns=["application_type"], axis=1, inplace=True)
df.shape
```

Out[ ]: (353446, 28)

In [ ]:
```
df.describe(include="object")
```

Out[ ]:

	term	home_ownership	verification_status	loan_status	purpose	pub_rec	initial_list_status	pub_rec_bankruptcies	emp_duration_type	issue_month	earliest_cr_month	state	mort_ac
count	353446	353446	353446	353446	353446	353446	353446	353446	353446	353446	353446	353446	35344
unique	2	6	3	2	14	2	2	2	3	12	12	54	
top	36	MORTGAGE	Verified	Fully Paid	debt_consolidation	good	f	good	equalsto	Oct	Oct	AP	ba
freq	272182	169856	120801	283519	210407	299877	215053	311009	215327	37697	34297	12757	22067

Data Preparation for modelling

- Scaling Numerical Columns

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

Out[ ]:

	int_rate	installment	grade	sub_grade	dti	open_acc	issue_year	earliest_cr_year	pincode	loan_amnt	annual_inc	emp_length	revo
count	353446.000000	353446.000000	353446.000000	353446.000000	353446.000000	353446.000000	353446.000000	353446.000000	353446.000000	353446.000000	353446.000000	353446.000000	353446.0
mean	13.661327	411.192627	2.820867	2.972726	17.234207	10.654128	2013.598185	1998.242088	34058.142556	13407.138431	65994.235085	5.947352	12611.7
std	4.432925	234.679481	1.322015	1.405934	8.020701	4.283042	1.464845	7.073114	25631.146185	7867.213303	29700.388927	3.442470	9112.2
min	5.320000	19.870000	1.000000	1.000000	0.000000	0.000000	2007.000000	1944.000000	813.000000	700.000000	4000.000000	1.000000	0.0
25%	10.640000	242.630000	2.000000	2.000000	11.260000	7.000000	2013.000000	1995.000000	11650.000000	7500.000000	44100.000000	3.000000	5700.0
50%	13.330000	361.380000	3.000000	3.000000	16.800000	10.000000	2014.000000	2000.000000	29597.000000	12000.000000	60000.000000	6.000000	10382.0
75%	16.490000	535.490000	4.000000	4.000000	22.800000	13.000000	2015.000000	2003.000000	48052.000000	18000.000000	82567.500000	10.000000	17476.0
max	30.990000	1479.490000	7.000000	5.000000	39.990000	23.000000	2016.000000	2013.000000	93700.000000	40000.000000	165300.000000	10.000000	42465.0

In [ ]:
```
# Standard Scaler
from sklearn.preprocessing import StandardScaler

numeric_columns = df.describe().columns
scaler = StandardScaler()
scaler.fit(df[numeric_columns])
df[numeric_columns] = scaler.transform(df[numeric_columns])
```

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

Out[ ]:

	term	int_rate	installment	grade	sub_grade	home_ownership	verification_status	loan_status	purpose	dti	open_acc	pub_rec	initial_list_status	pub_rec_bankruptcies	emp_dura
0	36	-0.501098	-0.348189	-0.620922	0.730671	RENT	Not Verified	Fully Paid	vacation	1.122820	1.248150	good	w	good	gr
1	36	-0.377026	-0.620049	-0.620922	1.441943	MORTGAGE	Not Verified	Fully Paid	debt_consolidation	0.600421	1.481629	good	f	good	
2	36	-0.715404	0.408121	-0.620922	0.019399	RENT	Source Verified	Fully Paid	credit_card	-0.554093	0.547712	good	f	good	
3	36	-1.617744	-0.811928	-1.377344	-0.691873	RENT	Not Verified	Fully Paid	credit_card	-1.824557	-1.086642	good	f	good	
4	60	0.814062	0.844290	0.135500	1.441943	MORTGAGE	Verified	Charged Off	credit_card	2.084084	0.547712	good	f	good	

- Encoding categorical features

1. For Label Encoding: term, loan_status, pub_rec, initial_list_status,pub_rec_bankruptcies, mort_acc

2. For OHE: verification_status, emp_duration_type

3. For Target Encoding: home_ownership, purpose, issue_month, earliest_cr_month, state

In [ ]:
```
# Label Encoding
from sklearn.preprocessing import LabelEncoder

labelencoder = LabelEncoder()
labelencoderlist = ["term", "loan_status", "pub_rec", "initial_list_status", "pub_rec_bankruptcies", "mort_acc"]

# Apply label encoding to each column
for column in labelencoderlist:
    df[column] = LabelEncoder().fit_transform(df[column])
```

In [ ]:
```
# OHE
ohe_list = ["verification_status", "emp_duration_type"]
df = pd.get_dummies(df,columns=ohe_list)
```

In [ ]:
```
ohe_encoded = ["verification_status_Not Verified", "verification_status_Source Verified", "verification_status_Verified"
               ,"emp_duration_type_equalsto", "emp_duration_type_greaterthan", "emp_duration_type_lessthan"]

for cols in ohe_encoded:
    df[cols] = df[cols].astype(int)
```

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

Out[ ]:

	term	int_rate	installment	grade	sub_grade	home_ownership	loan_status	purpose	dti	open_acc	pub_rec	initial_list_status	pub_rec_bankruptcies	issue_month	issue_year	earl
0	0	-0.501098	-0.348189	-0.620922	0.730671	RENT	1	vacation	1.122820	1.248150	1	1	1	Jan	0.956973	
1	0	-0.377026	-0.620049	-0.620922	1.441943	MORTGAGE	1	debt_consolidation	0.600421	1.481629	1	0	1	Jan	0.956973	
2	0	-0.715404	0.408121	-0.620922	0.019399	RENT	1	credit_card	-0.554093	0.547712	1	0	1	Jan	0.956973	
3	0	-1.617744	-0.811928	-1.377344	-0.691873	RENT	1	credit_card	-1.824557	-1.086642	1	0	1	Nov	0.274306	
4	1	0.814062	0.844290	0.135500	1.441943	MORTGAGE	0	credit_card	2.084084	0.547712	1	0	1	Apr	-0.408361	

In [ ]:
```
import category_encoders as ce

# Specify columns to target encode
targetencodelist = ["home_ownership", "purpose", "issue_month", "earliest_cr_month", "state"]

# Initialize TargetEncoder
target_encoder = ce.TargetEncoder(cols=targetencodelist)

# Fit and transform the selected columns
df[targetencodelist] = target_encoder.fit_transform(df[targetencodelist], df['loan_status'])
```

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

Out[ ]:

	term	int_rate	installment	grade	sub_grade	home_ownership	loan_status	purpose	dti	open_acc	pub_rec	initial_list_status	pub_rec_bankruptcies	issue_month	issue_year	earliest_cr_m
0	0	-0.501098	-0.348189	-0.620922	0.730671	0.774488	1	0.809965	1.122820	1.248150	1	1	1	0.799079	0.956973	0.80
1	0	-0.377026	-0.620049	-0.620922	1.441943	0.828625	1	0.791086	0.600421	1.481629	1	0	1	0.799079	0.956973	0.79
2	0	-0.715404	0.408121	-0.620922	0.019399	0.774488	1	0.831658	-0.554093	0.547712	1	0	1	0.799079	0.956973	0.80
3	0	-1.617744	-0.811928	-1.377344	-0.691873	0.774488	1	0.831658	-1.824557	-1.086642	1	0	1	0.813152	0.274306	0.80
4	1	0.814062	0.844290	0.135500	1.441943	0.828625	0	0.831658	2.084084	0.547712	1	0	1	0.792540	-0.408361	0.79

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

Out[ ]:
```
(353446, 32)
```

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

```
<class 'pandas.core.frame.DataFrame'>
Index: 353446 entries, 0 to 353830
Data columns (total 32 columns):
 #   Column                                Non-Null Count   Dtype  
---  ------                                --------------   -----  
 0   term                                  353446 non-null  int64  
 1   int_rate                              353446 non-null  float64
 2   installment                           353446 non-null  float64
 3   grade                                 353446 non-null  float64
 4   sub_grade                             353446 non-null  float64
 5   home_ownership                        353446 non-null  float64
 6   loan_status                           353446 non-null  int64  
 7   purpose                               353446 non-null  float64
 8   dti                                   353446 non-null  float64
 9   open_acc                              353446 non-null  float64
 10  pub_rec                               353446 non-null  int64  
 11  initial_list_status                   353446 non-null  int64  
 12  pub_rec_bankruptcies                  353446 non-null  int64  
 13  issue_month                           353446 non-null  float64
 14  issue_year                            353446 non-null  float64
 15  earliest_cr_month                     353446 non-null  float64
 16  earliest_cr_year                      353446 non-null  float64
 17  state                                 353446 non-null  float64
 18  pincode                               353446 non-null  float64
 19  loan_amnt                             353446 non-null  float64
 20  annual_inc                            353446 non-null  float64
 21  emp_length                            353446 non-null  float64
 22  mort_acc                              353446 non-null  int64  
 23  revol_bal                             353446 non-null  float64
 24  total_acc                             353446 non-null  float64
 25  revol_util                            353446 non-null  float64
 26  verification_status_Not Verified      353446 non-null  int64  
 27  verification_status_Source Verified   353446 non-null  int64  
 28  verification_status_Verified          353446 non-null  int64  
 29  emp_duration_type_equalsto            353446 non-null  int64  
 30  emp_duration_type_greaterthan         353446 non-null  int64  
 31  emp_duration_type_lessthan            353446 non-null  int64  
dtypes: float64(20), int64(12)
memory usage: 89.0 MB
```

Multicollinearity and Feature Selection Check

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

Out[ ]:

	term	int_rate	installment	grade	sub_grade	home_ownership	loan_status	purpose	dti	open_acc	pub_rec	initial_list_status	pub_rec_bankruptcies	issue_month	issue_year	earliest_cr_m
0	0	-0.501098	-0.348189	-0.620922	0.730671	0.774488	1	0.809965	1.122820	1.248150	1	1	1	0.799079	0.956973	0.8(
1	0	-0.377026	-0.620049	-0.620922	1.441943	0.828625	1	0.791086	0.600421	1.481629	1	0	1	0.799079	0.956973	0.79
2	0	-0.715404	0.408121	-0.620922	0.019399	0.774488	1	0.831658	-0.554093	0.547712	1	0	1	0.799079	0.956973	0.8(
3	0	-1.617744	-0.811928	-1.377344	-0.691873	0.774488	1	0.831658	-1.824557	-1.086642	1	0	1	0.813152	0.274306	0.8(
4	1	0.814062	0.844290	0.135500	1.441943	0.828625	0	0.831658	2.084084	0.547712	1	0	1	0.792540	-0.408361	0.79

In [ ]:
```
# VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = pd.DataFrame()
vif['Features'] = df.columns
vif['VIF'] = [variance_inflation_factor(df.values, i) for i in range(df.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

```
/home/varun/Documents/workspace/Neoversity/Business Cases/MicroFinancing/.venv/lib/python3.10/site-packages/statsmodels/stats/outliers_influence.py:198: RuntimeWarning: divide by zero encountered in scalar divide
  vif = 1. / (1. - r_squared_i)
```

Out[ ]:

	Features	VIF
30	emp_duration_type_greaterthan	inf
31	emp_duration_type_lessthan	inf
27	verification_status_Source Verified	inf
26	verification_status_Not Verified	inf
29	emp_duration_type_equalsto	inf
28	verification_status_Verified	inf
19	loan_amnt	59.52
2	installment	50.92
3	grade	23.80
1	int_rate	23.14
0	term	6.84
10	pub_rec	4.36
12	pub_rec_bankruptcies	4.28
21	emp_length	3.39
4	sub_grade	2.05
24	total_acc	2.03
9	open_acc	2.01
23	revol_bal	1.99
22	mort_acc	1.83
5	home_ownership	1.67
20	annual_inc	1.66
25	revol_util	1.58
14	issue_year	1.53
8	dti	1.45
11	initial_list_status	1.27
16	earliest_cr_year	1.24
6	loan_status	1.24
18	pincode	1.14
7	purpose	1.04
13	issue_month	1.03
15	earliest_cr_month	1.00
17	state	1.00

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

Out[ ]:

	term	int_rate	installment	grade	sub_grade	home_ownership	loan_status	purpose	dti	open_acc	pub_rec	initial_list_status	pub_rec_bankruptcies	issue_month	issue_year	earliest_cr_m
0	0	-0.501098	-0.348189	-0.620922	0.730671	0.774488	1	0.809965	1.122820	1.248150	1	1	1	0.799079	0.956973	0.80
1	0	-0.377026	-0.620049	-0.620922	1.441943	0.828625	1	0.791086	0.600421	1.481629	1	0	1	0.799079	0.956973	0.79
2	0	-0.715404	0.408121	-0.620922	0.019399	0.774488	1	0.831658	-0.554093	0.547712	1	0	1	0.799079	0.956973	0.80
3	0	-1.617744	-0.811928	-1.377344	-0.691873	0.774488	1	0.831658	-1.824557	-1.086642	1	0	1	0.813152	0.274306	0.80
4	1	0.814062	0.844290	0.135500	1.441943	0.828625	0	0.831658	2.084084	0.547712	1	0	1	0.792540	-0.408361	0.79

In [ ]:
```
df.to_csv("dataset/df_processed_phase2.csv", index=False)
```