In [3]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [5]:
```
#uploadind data file
loan = pd.read_csv(r"C:\Users\dell\Downloads\loan (1).zip")
```

In [6]:
```
loan.head(10)
```

Out[6]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_gr
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	
5	1075269	1311441	5000	5000	5000.0	36 months	7.90%	156.46	A	
6	1069639	1304742	7000	7000	7000.0	60 months	15.96%	170.08	C	
7	1072053	1288686	3000	3000	3000.0	36 months	18.64%	109.43	E	
8	1071795	1306957	5600	5600	5600.0	60 months	21.28%	152.39	F	
9	1071570	1306721	5375	5375	5350.0	60 months	12.69%	121.45	B	

10 rows × 111 columns

In [7]:
```
loan.shape
```

Out[7]:
```
(39717, 111)
```

In [8]:
```
loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39717 entries, 0 to 39716
Columns: 111 entries, id to total_il_high_credit_limit
dtypes: float64(74), int64(13), object(24)
memory usage: 33.6+ MB
```

In [9]:
```
loan.isnull().any()
```

Out[9]:
```
id                          False
member_id                   False
loan_amnt                   False
funded_amnt                 False
funded_amnt_inv             False
                            ...
tax_liens                    True
tot_hi_cred_lim              True
total_bal_ex_mort            True
total_bc_limit               True
total_il_high_credit_limit   True
Length: 111, dtype: bool
```

In [10]:
```
loan.isnull().any().sum()
```

Out[10]:
```
68
```

In [11]:
```
loan.describe()
```

Out[11]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	installment	annual_inc
count	3.971700e+04	3.971700e+04	39717.000000	39717.000000	39717.000000	39717.000000	3.971700e+04
mean	6.831319e+05	8.504636e+05	11219.443815	10947.713196	10397.448868	324.561922	6.896893e+04
std	2.106941e+05	2.656783e+05	7456.670694	7187.238670	7128.450439	208.874874	6.379377e+04
min	5.473400e+04	7.069900e+04	500.000000	500.000000	0.000000	15.690000	4.000000e+03
25%	5.162210e+05	6.667800e+05	5500.000000	5400.000000	5000.000000	167.020000	4.040400e+04
50%	6.656650e+05	8.508120e+05	10000.000000	9600.000000	8975.000000	280.220000	5.900000e+04
75%	8.377550e+05	1.047339e+06	15000.000000	15000.000000	14400.000000	430.780000	8.230000e+04
max	1.077501e+06	1.314167e+06	35000.000000	35000.000000	35000.000000	1305.190000	6.000000e+06

8 rows × 87 columns

In [12]:
```
loan.describe(include = 'all')
```

Out[12]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installme
count	3.971700e+04	3.971700e+04	39717.000000	39717.000000	39717.000000	39717	39717	39717.0000
unique	NaN	NaN	NaN	NaN	NaN	2	371	Na
top	NaN	NaN	NaN	NaN	NaN	36 months	10.99%	Na
freq	NaN	NaN	NaN	NaN	NaN	29096	956	Na
mean	6.831319e+05	8.504636e+05	11219.443815	10947.713196	10397.448868	NaN	NaN	324.5619
std	2.106941e+05	2.656783e+05	7456.670694	7187.238670	7128.450439	NaN	NaN	208.8748
min	5.473400e+04	7.069900e+04	500.000000	500.000000	0.000000	NaN	NaN	15.6900
25%	5.162210e+05	6.667800e+05	5500.000000	5400.000000	5000.000000	NaN	NaN	167.0200
50%	6.656650e+05	8.508120e+05	10000.000000	9600.000000	8975.000000	NaN	NaN	280.2200
75%	8.377550e+05	1.047339e+06	15000.000000	15000.000000	14400.000000	NaN	NaN	430.7800
max	1.077501e+06	1.314167e+06	35000.000000	35000.000000	35000.000000	NaN	NaN	1305.1900

11 rows × 111 columns

In [13]:
```
columns = loan.columns
```

In [14]:
```
percent_missing = round(100*loan.isnull().sum()/len(loan.index),2)
```

In [15]:
```
missing_val_df = pd.DataFrame({'column_name': columns , 'percent_missing':percent_missin
```

In [16]:
```
drop_missing = missing_val_df[missing_val_df.percent_missing == 100].column_name
```

In [17]:
```
loan = loan.drop(drop_missing, axis = 1)
```

In [18]:
```
loan
```

Out[18]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	su
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	
...	...	...	...	...	...	...	...	...	...	
39712	92187	92174	2500	2500	1075.0	36 months	8.07%	78.42	A	
39713	90665	90607	8500	8500	875.0	36 months	10.28%	275.38	C	
39714	90395	90390	5000	5000	1325.0	36 months	8.07%	156.84	A	
39715	90376	89243	5000	5000	650.0	36 months	7.43%	155.38	A	
39716	87023	86999	7500	7500	800.0	36 months	13.75%	255.43	E	

39717 rows × 57 columns

In [19]:
```
loan.columns
```

Out[19]:
```
Index(['id', 'member_id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv',
       'term', 'int_rate', 'installment', 'grade', 'sub_grade', 'emp_title',
       'emp_length', 'home_ownership', 'annual_inc', 'verification_status',
       'issue_d', 'loan_status', 'pymnt_plan', 'url', 'desc', 'purpose',
       'title', 'zip_code', 'addr_state', 'dti', 'delinq_2yrs',
       'earliest_cr_line', 'inq_last_6mths', 'mths_since_last_delinq',
       'mths_since_last_record', 'open_acc', 'pub_rec', 'revol_bal',
       'revol_util', 'total_acc', 'initial_list_status', 'out_prncp',
       'out_prncp_inv', 'total_pymnt', 'total_pymnt_inv', 'total_rec_prncp',
       'total_rec_int', 'total_rec_late_fee', 'recoveries',
       'collection_recovery_fee', 'last_pymnt_d', 'last_pymnt_amnt',
       'next_pymnt_d', 'last_credit_pull_d', 'collections_12_mths_ex_med',
       'policy_code', 'application_type', 'acc_now_delinq',
       'chargeoff_within_12_mths', 'delinq_amnt', 'pub_rec_bankruptcies',
       'tax_liens'],
      dtype='object')
```

In [20]:
```
#Personal Informtion of customers, and is not usefull for EDA purpose

loan.drop(['title', 'emp_title','url', 'zip_code', 'desc'], axis = 1, inplace = True)
```

In [21]:
```
loan.drop(['last_credit_pull_d','last_pymnt_amnt','last_pymnt_d','collection_recovery_fe
          'total_rec_late_fee', 'total_rec_int','total_rec_prncp', 'total_pymnt_inv', 'o

loan.shape
```

Out[21]:
```
(39717, 41)
```

In [22]:
```
loan.columns
```

Out[22]:
```
Index(['id', 'member_id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv',
       'term', 'int_rate', 'installment', 'grade', 'sub_grade', 'emp_length',
       'home_ownership', 'annual_inc', 'verification_status', 'issue_d',
       'loan_status', 'pymnt_plan', 'purpose', 'addr_state', 'dti',
       'delinq_2yrs', 'earliest_cr_line', 'inq_last_6mths',
       'mths_since_last_delinq', 'mths_since_last_record', 'open_acc',
       'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
       'initial_list_status', 'total_pymnt', 'next_pymnt_d',
       'collections_12_mths_ex_med', 'policy_code', 'application_type',
       'acc_now_delinq', 'chargeoff_within_12_mths', 'delinq_amnt',
       'pub_rec_bankruptcies', 'tax_liens'],
      dtype='object')
```

In [23]:
```
loan.drop(['installment','pymnt_plan',
       'delinq_2yrs', 'earliest_cr_line', 'inq_last_6mths',
       'mths_since_last_delinq', 'mths_since_last_record', 'open_acc',
       'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
       'initial_list_status', 'total_pymnt', 'next_pymnt_d',
       'collections_12_mths_ex_med', 'policy_code', 'application_type',
       'acc_now_delinq', 'chargeoff_within_12_mths', 'delinq_amnt',
       'pub_rec_bankruptcies', 'tax_liens'], axis=1, inplace = True)
```

In [24]:
```
loan.shape
```

Out[24]:
```
(39717, 18)
```

In [25]:
```
loan.columns
```

Out[25]:
```
Index(['id', 'member_id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv',
       'term', 'int_rate', 'grade', 'sub_grade', 'emp_length',
       'home_ownership', 'annual_inc', 'verification_status', 'issue_d',
       'loan_status', 'purpose', 'addr_state', 'dti'],
      dtype='object')
```

In [26]:
```
#Check if the there is any repeated loan ID
loan['id'].duplicated().sum()
```

Out[26]:
```
0
```

In [27]:
```
#Reoving id's, they are'nt useful for analysis.
loan.drop(['id', 'member_id'], axis = 1, inplace = True)
```

In [28]:
```
loan.shape
```

Out[28]:
```
(39717, 16)
```

In [29]:
```
loan.head()
```

Out[29]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	grade	sub_grade	emp_length	home_ownership
0	5000	5000	4975.0	36 months	10.65%	B	B2	10+ years	RENT
1	2500	2500	2500.0	60 months	15.27%	C	C4	< 1 year	RENT
2	2400	2400	2400.0	36 months	15.96%	C	C5	10+ years	RENT
3	10000	10000	10000.0	36 months	13.49%	C	C1	10+ years	RENT
4	3000	3000	3000.0	60 months	12.69%	B	B5	1 year	RENT

Since we will be analysing the scenario where the chances are more that a loan applicant will default. Hence, we will ignore the loans that are currently running and therefore, we will use loans which are fully paid and charged off for the analysis.

In [30]:
```
loan = loan[loan.loan_status != 'Current']
```

In [31]:
```
#Checking the null values
round(100*loan.isnull().sum()/len(loan.index),2)
```

Out[31]:
```
loan_amnt              0.00
funded_amnt            0.00
funded_amnt_inv        0.00
term                   0.00
int_rate               0.00
grade                  0.00
sub_grade              0.00
emp_length             2.68
home_ownership         0.00
annual_inc             0.00
verification_status    0.00
issue_d                0.00
loan_status            0.00
purpose                0.00
addr_state             0.00
dti                    0.00
dtype: float64
```

In [32]:
```
#Removing the null values from emp_length as the null percentage is less than 5%, hence
loan = loan[~loan['emp_length'].isnull()]
```

checking the purpose of loan as this is the important factor to be used in analysis.

In [33]:
```
round(loan.purpose.value_counts(normalize = True)*100, 2)
```

Out[33]:
```
debt_consolidation    47.08
credit_card           13.05
other                  9.89
home_improvement       7.42
major_purchase         5.54
small_business         4.55
car                    3.86
wedding                2.43
medical                1.75
moving                 1.47
house                  0.94
vacation               0.93
educational            0.84
renewable_energy       0.25
Name: purpose, dtype: float64
```

In [34]:
```
#Since, we do not know what the terms other stands for, so get ris of it.
loan.drop(loan[loan.purpose == 'other'].index, inplace = True)
```

In [35]:
```
loan.purpose.value_counts()
```

Out[35]:
```
debt_consolidation    17675
credit_card            4899
home_improvement       2785
major_purchase         2080
small_business         1710
car                    1448
wedding                 913
medical                 656
moving                  552
house                   354
vacation                348
educational             317
renewable_energy         94
Name: purpose, dtype: int64
```

In [36]:
```
loan.term.value_counts()
```

Out[36]:
```
 36 months    25270
 60 months     8561
Name: term, dtype: int64
```

In [37]:
```
loan.term = loan.term.apply(lambda x: int(x.split()[0]) if x.find('month')>0  else int(x

loan.head(5)
```

Out[37]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	grade	sub_grade	emp_length	home_ownership
0	5000	5000	4975.0	36	10.65%	B	B2	10+ years	RENT
1	2500	2500	2500.0	60	15.27%	C	C4	< 1 year	RENT
2	2400	2400	2400.0	36	15.96%	C	C5	10+ years	RENT
5	5000	5000	5000.0	36	7.90%	A	A4	3 years	RENT
6	7000	7000	7000.0	60	15.96%	C	C5	8 years	RENT

In [38]:
```
loan['int_rate'].value_counts()
```

Out[38]:
```
10.99%    800
11.49%    685
13.49%    669
7.51%     660
7.88%     637
          ...
16.20%      1
18.72%      1
16.01%      1
16.96%      1
14.70%      1
Name: int_rate, Length: 367, dtype: int64
```

In [39]:
```
loan['int_rate'] = loan['int_rate'].str.replace('%','')

loan['int_rate'] = loan['int_rate'].astype(float)

loan.dtypes
```

Out[39]:
```
loan_amnt               int64
funded_amnt             int64
funded_amnt_inv       float64
term                    int64
int_rate              float64
grade                  object
sub_grade              object
emp_length             object
home_ownership         object
annual_inc            float64
verification_status    object
issue_d                object
loan_status            object
purpose                object
addr_state             object
dti                   float64
dtype: object
```

In [40]:
```
loan.head(5)
```

Out[40]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	grade	sub_grade	emp_length	home_ownership
0	5000	5000	4975.0	36	10.65	B	B2	10+ years	RENT
1	2500	2500	2500.0	60	15.27	C	C4	< 1 year	RENT
2	2400	2400	2400.0	36	15.96	C	C5	10+ years	RENT
5	5000	5000	5000.0	36	7.90	A	A4	3 years	RENT
6	7000	7000	7000.0	60	15.96	C	C5	8 years	RENT

In [41]:
```
loan.emp_length.value_counts()
```

Out[41]:
```
10+ years    7664
< 1 year     3994
2 years      3847
3 years      3636
4 years      3027
5 years      2895
1 year       2811
6 years      1987
7 years      1550
8 years      1308
9 years      1112
Name: emp_length, dtype: int64
```

In [42]:
```
#Cleaning the emp_length and converting it to int
#Should be treating 0 and <1 as 0 as they all fall under less than 1 year category
#Should be treating 10+ years as 10 which would mean 10 years and above
loan['emp_length'] = loan['emp_length'].str.replace('+', '')
loan['emp_length'] =loan['emp_length'].str.replace('< 1 ', '0')
loan['emp_length'] =loan['emp_length'].str.replace('years','')
loan['emp_length'] =loan['emp_length'].str.replace('year','')
loan['emp_length'] = loan['emp_length'].astype(int)
```

In [43]:
```
loan.emp_length.value_counts()
```

Out[43]:
```
10    7664
0     3994
2     3847
3     3636
4     3027
5     2895
1     2811
6     1987
7     1550
8     1308
9     1112
Name: emp_length, dtype: int64
```

In [44]:
```
loan.head(5)
```

Out[44]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	grade	sub_grade	emp_length	home_ownership
0	5000	5000	4975.0	36	10.65	B	B2	10	RENT
1	2500	2500	2500.0	60	15.27	C	C4	0	RENT
2	2400	2400	2400.0	36	15.96	C	C5	10	RENT
5	5000	5000	5000.0	36	7.90	A	A4	3	RENT
6	7000	7000	7000.0	60	15.96	C	C5	8	RENT

In [45]:
```
loan['annual_inc'].describe()
```

Out[45]:
```
count    3.383100e+04
mean     7.003334e+04
std      6.596019e+04
min      4.000000e+03
25%      4.200000e+04
50%      6.000000e+04
75%      8.400000e+04
max      6.000000e+06
Name: annual_inc, dtype: float64
```

In [46]:
```
loan['annual_inc'].quantile([0.5, 0.7,0.9, 0.95, 0.99, 1])
```

Out[46]:
```
0.50      60000.0
0.70      77000.0
0.90     118000.0
0.95     143387.5
0.99     235560.0
1.00    6000000.0
Name: annual_inc, dtype: float64
```

In [47]:
```
quartile = loan['annual_inc'].quantile(0.99)
loan = loan[loan['annual_inc'] < quartile]

loan['annual_inc'].describe()
```

Out[47]:
```
count     33492.000000
mean      66490.324304
std       35165.906151
min        4000.000000
25%       42000.000000
50%       60000.000000
75%       82000.000000
max      235000.000000
Name: annual_inc, dtype: float64
```

In [48]:
```
loan['issue_d'].value_counts()
```

Out[48]:
```
Dec-11    1853
Nov-11    1791
Sep-11    1670
Oct-11    1661
Aug-11    1569
Jun-11    1479
Jul-11    1474
May-11    1419
Apr-11    1371
Mar-11    1304
Jan-11    1244
Feb-11    1154
Dec-10    1116
Oct-10    1004
Nov-10    1000
Jul-10     971
Sep-10     955
Aug-10     944
Jun-10     850
May-10     771
Apr-10     667
Mar-10     599
Feb-10     506
Nov-09     504
Jan-10     484
Dec-09     473
Oct-09     464
Sep-09     407
Aug-09     363
Jul-09     331
Jun-09     318
May-09     283
Mar-09     251
Apr-09     246
Feb-09     228
Jan-09     221
Dec-08     203
Mar-08     198
Nov-08     167
Feb-08     154
Jan-08     145
Apr-08     125
Oct-08      82
Dec-07      74
Jul-08      65
May-08      62
Aug-08      57
Jun-08      56
Oct-07      31
Nov-07      29
Aug-07      29
Jul-07      28
Sep-08      27
Sep-07      14
Jun-07       1
Name: issue_d, dtype: int64
```

In [49]:
```
loan['issue_month'] = loan['issue_d'].apply(lambda x: str(x.split('-')[0])if x.find('-')

loan['issue_year'] = loan['issue_d'].apply(lambda x: str(x.split('-')[1])if x.find('-')

loan['issue_year'] = '20'+ loan['issue_year']

loan.head(5)
```

Out[49]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	grade	sub_grade	emp_length	home_ownership
0	5000	5000	4975.0	36	10.65	B	B2	10	RENT
1	2500	2500	2500.0	60	15.27	C	C4	0	RENT
2	2400	2400	2400.0	36	15.96	C	C5	10	RENT
5	5000	5000	5000.0	36	7.90	A	A4	3	RENT
6	7000	7000	7000.0	60	15.96	C	C5	8	RENT

In [50]:
```
loan['loan_inc_ratio'] = round(100*(loan['loan_amnt']/loan['annual_inc']),2)
loan.head(2)
```

Out[50]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	grade	sub_grade	emp_length	home_ownership
0	5000	5000	4975.0	36	10.65	B	B2	10	RENT
1	2500	2500	2500.0	60	15.27	C	C4	0	RENT

In [51]:
```
loan['loan_inc_ratio'].quantile([.25,.50,.75])
```

Out[51]:
```
0.25    10.29
0.50    16.67
0.75    25.38
Name: loan_inc_ratio, dtype: float64
```

In [52]:
```
# Categorising loan_inc_ratio into categorised_loan_inc_ratio column
# < 10 is low
# between 10 and 16( both inclusive) is medium
# between 17 and 24 is high
# greater than 24 is very high

def loan_inc_ratio_category(n):
    if n < 10:
        return 'low'
    elif n >= 10 and n < 17:
        return 'medium'
    elif n >= 17 and n < 25:
        return 'high'
    else:
        return 'very high'
loan['categorised_loan_inc_ratio'] = loan['loan_inc_ratio'].apply(loan_inc_ratio_categor
loan.head()
```

Out[52]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	grade	sub_grade	emp_length	home_ownership
0	5000	5000	4975.0	36	10.65	B	B2	10	RENT
1	2500	2500	2500.0	60	15.27	C	C4	0	RENT
2	2400	2400	2400.0	36	15.96	C	C5	10	RENT
5	5000	5000	5000.0	36	7.90	A	A4	3	RENT
6	7000	7000	7000.0	60	15.96	C	C5	8	RENT

In [53]:
```
loan['int_rate'].quantile([.25,.5,.75])
```

Out[53]:
```
0.25     8.94
0.50    11.83
0.75    14.46
Name: int_rate, dtype: float64
```

In [54]:
```
#categorise int_rate column into categorised_int_rate_perc column
# < 9% is low
# between 9% and 11% ( both inclusive ) is medium
# between 12% to 13% is high
# Greater than 14% is very high

def interest_rates(n):
    if n < 9:
        return 'low'
    elif n >= 9 and n < 12:
        return 'medium'
    elif n >= 12 and n < 14:
        return 'high'
    else:

        return 'very high'
loan['categorised_int_rate_perc'] = loan['int_rate'].apply(interest_rates)
loan.head()
```

Out[54]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	grade	sub_grade	emp_length	home_ownership
0	5000	5000	4975.0	36	10.65	B	B2	10	RENT
1	2500	2500	2500.0	60	15.27	C	C4	0	RENT
2	2400	2400	2400.0	36	15.96	C	C5	10	RENT
5	5000	5000	5000.0	36	7.90	A	A4	3	RENT
6	7000	7000	7000.0	60	15.96	C	C5	8	RENT

5 rows × 21 columns

In [55]:
```
loan['emp_length'].quantile([.25,.5,.75])
```

Out[55]:
```
0.25    2.0
0.50    4.0
0.75    9.0
Name: emp_length, dtype: float64
```

In [56]:
```
#categorising emp_length column into categorised_emp_length
# < 2 is entry level
# between 2 and 4 (both inclusive ) is junior level
# between 5 and 9 is middle level
# Greater than 9 is senior level


def length_of_emp(n):
    if n < 2:
        return 'entry level'
    elif n >= 2 and n < 4:
        return 'junior level'
    elif n >= 4 and n < 9:
        return 'middle level'
    else:
        return 'senior level'
loan['categorised_emp_length'] = loan['emp_length'].apply(length_of_emp)
loan.head()
```

Out[56]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	grade	sub_grade	emp_length	home_ownership
0	5000	5000	4975.0	36	10.65	B	B2	10	RENT
1	2500	2500	2500.0	60	15.27	C	C4	0	RENT
2	2400	2400	2400.0	36	15.96	C	C5	10	RENT
5	5000	5000	5000.0	36	7.90	A	A4	3	RENT
6	7000	7000	7000.0	60	15.96	C	C5	8	RENT

5 rows × 22 columns

In [57]:
```
loan['annual_inc'].quantile([.25, .5, .75])
```

Out[57]:
```
0.25    42000.0
0.50    60000.0
0.75    82000.0
Name: annual_inc, dtype: float64
```

In [58]:
```
#categorising annual_inc column into categorised_annual_inc
# < 41000 is low inc
# between 41000 and 59000 (both inclusive ) is medium inc
# between 60000 and 82000 is high inc
# Greater than 820000 is very high inc

def annual_income(n):
    if n < 41000:
        return 'low income'
    elif n >= 41000 and n < 60000:
        return 'medium income'
    elif n >= 60000 and n < 83000:
        return 'high income'
    else:
        return 'very high income'
loan['categorised_annual_inc'] = loan['annual_inc'].apply(annual_income)
loan.head()
```

Out[58]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	grade	sub_grade	emp_length	home_ownership
0	5000	5000	4975.0	36	10.65	B	B2	10	RENT
1	2500	2500	2500.0	60	15.27	C	C4	0	RENT
2	2400	2400	2400.0	36	15.96	C	C5	10	RENT
5	5000	5000	5000.0	36	7.90	A	A4	3	RENT
6	7000	7000	7000.0	60	15.96	C	C5	8	RENT

5 rows × 23 columns

In [59]:
```
loan.categorised_annual_inc.isnull().sum()
```

Out[59]:
```
0
```

In [60]:
```
loan['dti'].quantile([.25, .5, .75])
```

Out[60]:
```
0.25     8.4075
0.50    13.5400
0.75    18.6500
Name: dti, dtype: float64
```

In [120…]:
```
loan['loan_amnt'].quantile([.25, .5, .75])
```

Out[120]:
```
0.25     6000.0
0.50    10000.0
0.75    15000.0
Name: loan_amnt, dtype: float64
```

In [63]:
```
#categorising loan_amnt into categorised loan_amnt
# < 5400 is low'
# between 5400 and 9600 (both inclusive) is medium
# between 9600 and 15000 is high
# > 15000 is very high

def loan_ammount(n):
    if n < 5400:
        return 'low'
    elif n >= 5400 and n < 9600:
        return 'medium'
    elif n >= 9600 and n < 15000:
        return 'high'
    else:
        return 'very high'
loan['categorised_loan_amnt'] = loan['loan_amnt'].apply(loan_ammount)
loan.head()
```

Out[63]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	grade	sub_grade	emp_length	home_ownership
0	5000	5000	4975.0	36	10.65	B	B2	10	RENT
1	2500	2500	2500.0	60	15.27	C	C4	0	RENT
2	2400	2400	2400.0	36	15.96	C	C5	10	RENT
5	5000	5000	5000.0	36	7.90	A	A4	3	RENT
6	7000	7000	7000.0	60	15.96	C	C5	8	RENT

5 rows × 24 columns

In [64]:
```
loan.loan_status.describe()
```

Out[64]:
```
count          33492
unique             2
top       Fully Paid
freq           28724
Name: loan_status, dtype: object
```

In [65]:
```
loan.columns
```

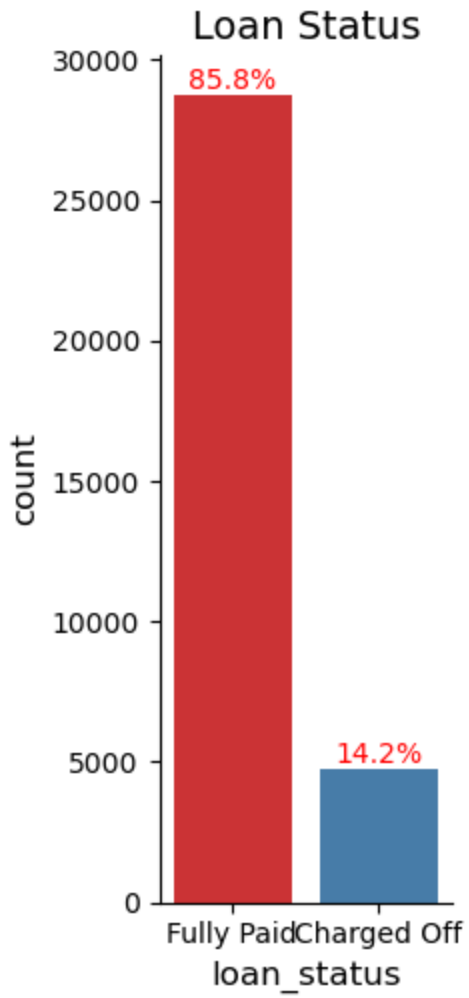
Out[65]:
```
Index(['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'term', 'int_rate',
       'grade', 'sub_grade', 'emp_length', 'home_ownership', 'annual_inc',
       'verification_status', 'issue_d', 'loan_status', 'purpose',
       'addr_state', 'dti', 'issue_month', 'issue_year', 'loan_inc_ratio',
       'categorised_loan_inc_ratio', 'categorised_int_rate_perc',
       'categorised_emp_length', 'categorised_annual_inc',
       'categorised_loan_amnt'],
      dtype='object')
```

In [66]:
```
loan.loan_status.value_counts(normalize =True)*100
```

Out[66]:
```
Fully Paid     85.763764
Charged Off    14.236236
Name: loan_status, dtype: float64
```

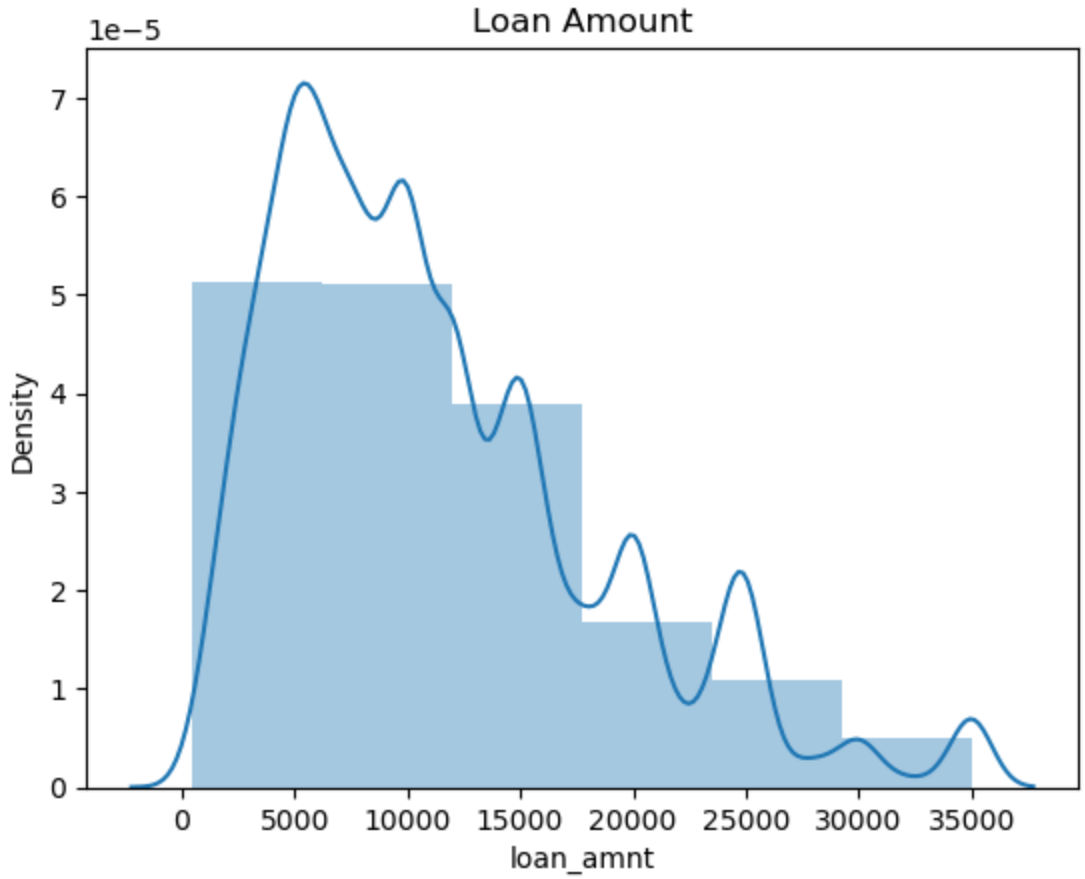
In [67]:
```
plot = sns.catplot(data = loan, x = 'loan_status', kind = 'count', palette ='Set1', aspe
an Status', fontsize = 14)
```

```
plt.xlabel('loan_status', fontsize = 12)
plt.ylabel('count', fontsize = 12)

#print the counts
ax = plot.facet_axis(0,0)
for p in ax.patches:
     ax.annotate('{:1.1f}%'.format((p.get_height()*100/len(loan))), ((p.get_x()+ p.get_w
                 color = 'red', ha = 'center', va = 'bottom')
plt.show()
```



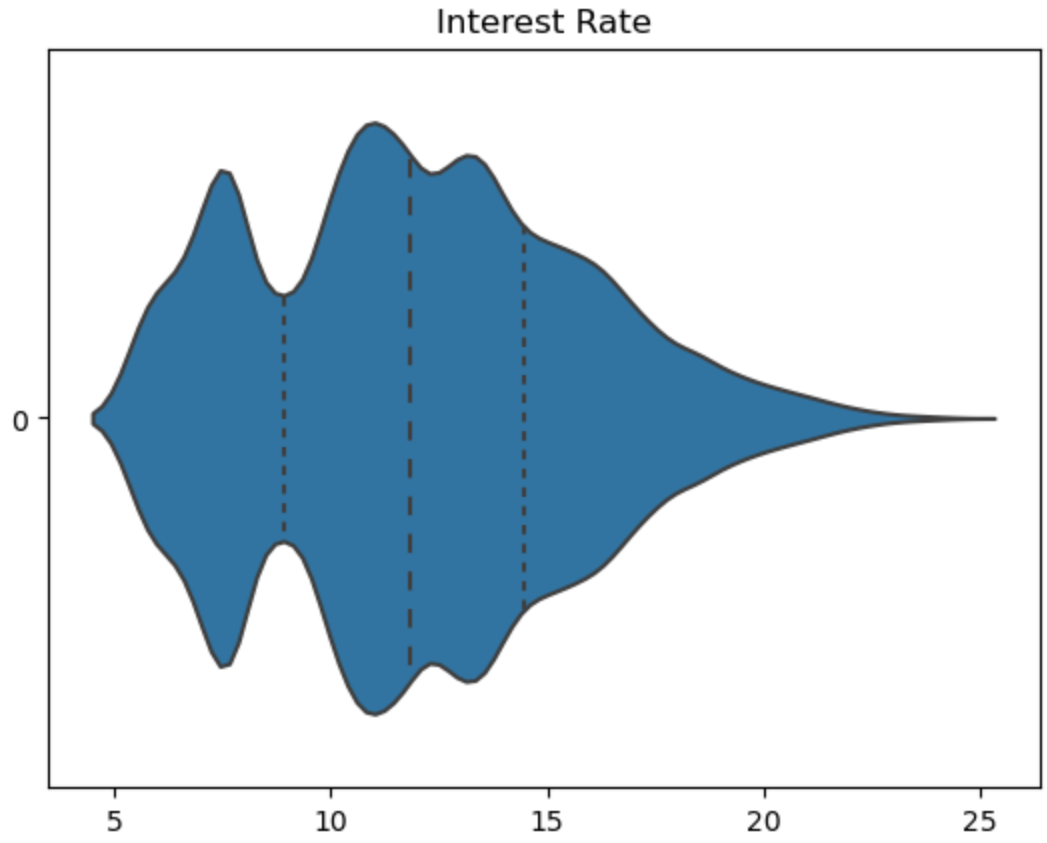
The above graph shows that in the data provided there are 14.2% applicants who have defaulted/charged off

In [68]:
```
sns.distplot(loan['loan_amnt'], bins = 6)
plt.title('Loan Amount')
plt.show()
```



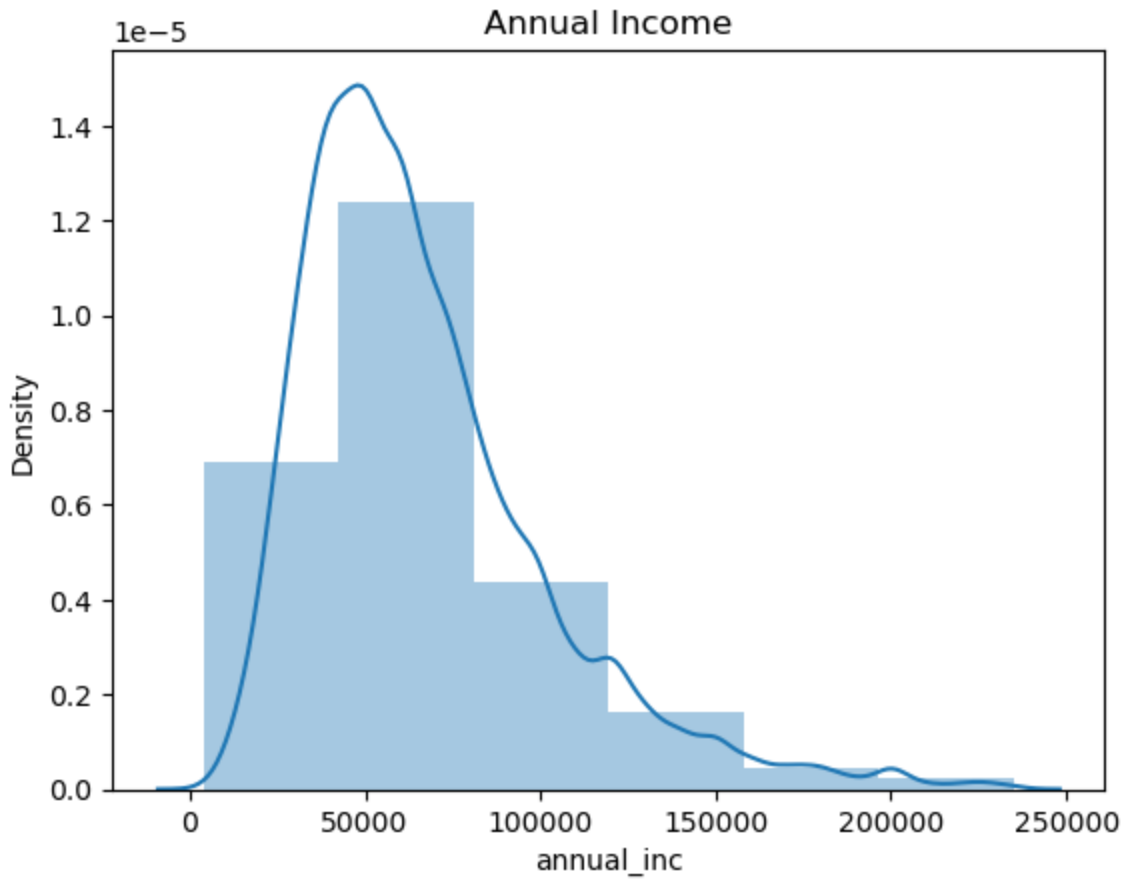
The graph shows that the loan amount is majorly spread around 6000 t0 16000 approx.

In [69]:
```
sns.violinplot(loan['int_rate'], inner = 'quartile', orient='h')
plt.title('Interest Rate')
plt.show()
```



The above graph shows that the loan percentage is spread around 8% to 14% approx.

In [70]:
```
sns.distplot(loan['annual_inc'], bins = 6)
plt.title('Annual Income')
plt.show()
```

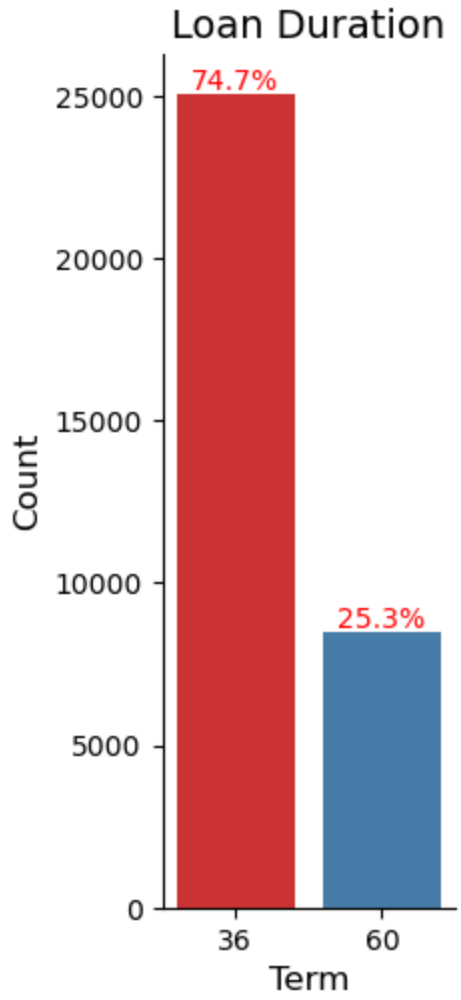


The above figure shows that the majority of applicants have annual income is spread around 40000USD to 90000USD

In [71]:
```
loan.columns
```

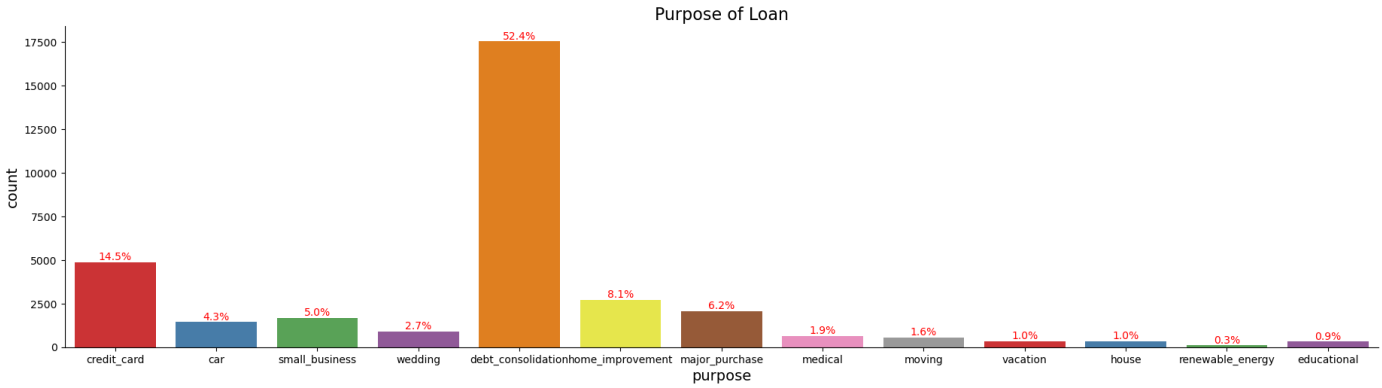
Out[71]:
```
Index(['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'term', 'int_rate',
       'grade', 'sub_grade', 'emp_length', 'home_ownership', 'annual_inc',
       'verification_status', 'issue_d', 'loan_status', 'purpose',
       'addr_state', 'dti', 'issue_month', 'issue_year', 'loan_inc_ratio',
       'categorised_loan_inc_ratio', 'categorised_int_rate_perc',
       'categorised_emp_length', 'categorised_annual_inc',
       'categorised_loan_amnt'],
      dtype='object')
```

In [72]:
```
plot = sns.catplot(data = loan, x = 'term', kind = 'count', palette ='Set1', aspect = .5
plt.title('Loan Duration', fontsize = 14)
plt.xlabel('Term', fontsize = 12)
plt.ylabel('Count', fontsize = 12)



#print the counts
ax = plot.facet_axis(0,0)
for p in ax.patches:
     ax.annotate('{:1.1f}%'.format((p.get_height()*100/len(loan))), ((p.get_x()+ p.get_w
                 color = 'red', ha = 'center', va = 'bottom')
plt.show()
```



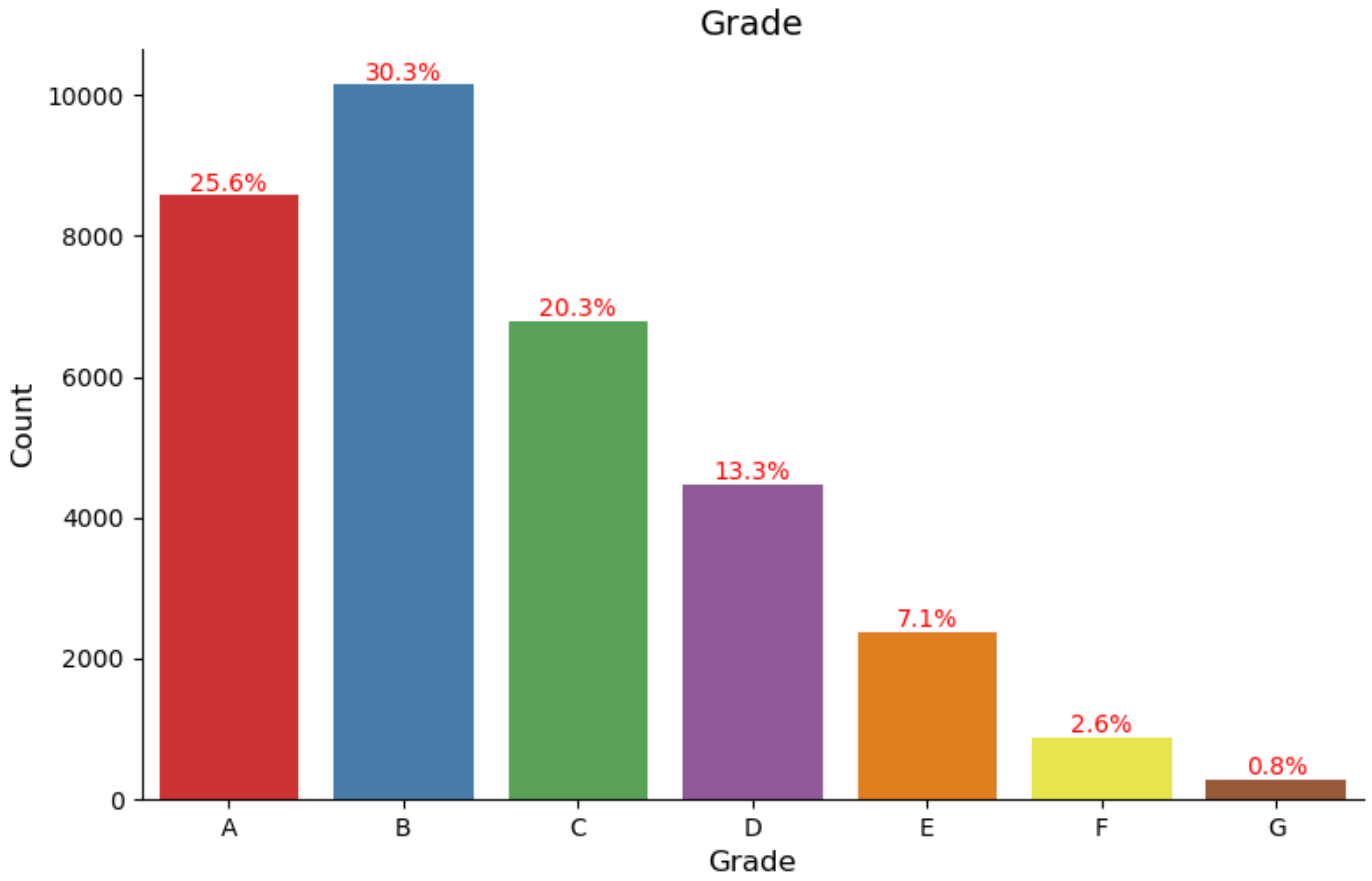
The above graph shows that 74.75 applicants have taken loan of 36 months duration

In [73]:
```
plot = sns.catplot(data = loan , x = 'purpose',kind = 'count', palette = 'Set1', aspect
plt.title('Purpose of Loan', fontsize = 16)
plt.xlabel('purpose', fontsize = 14)
plt.ylabel('count', fontsize = 14)

ax = plot.facet_axis(0,0)
for p in ax.patches:
     ax.annotate('{:1.1f}%'.format((p.get_height()*100/len(loan))), ((p.get_x()+ p.get_w
                 color = 'red', ha = 'center', va = 'bottom')
```



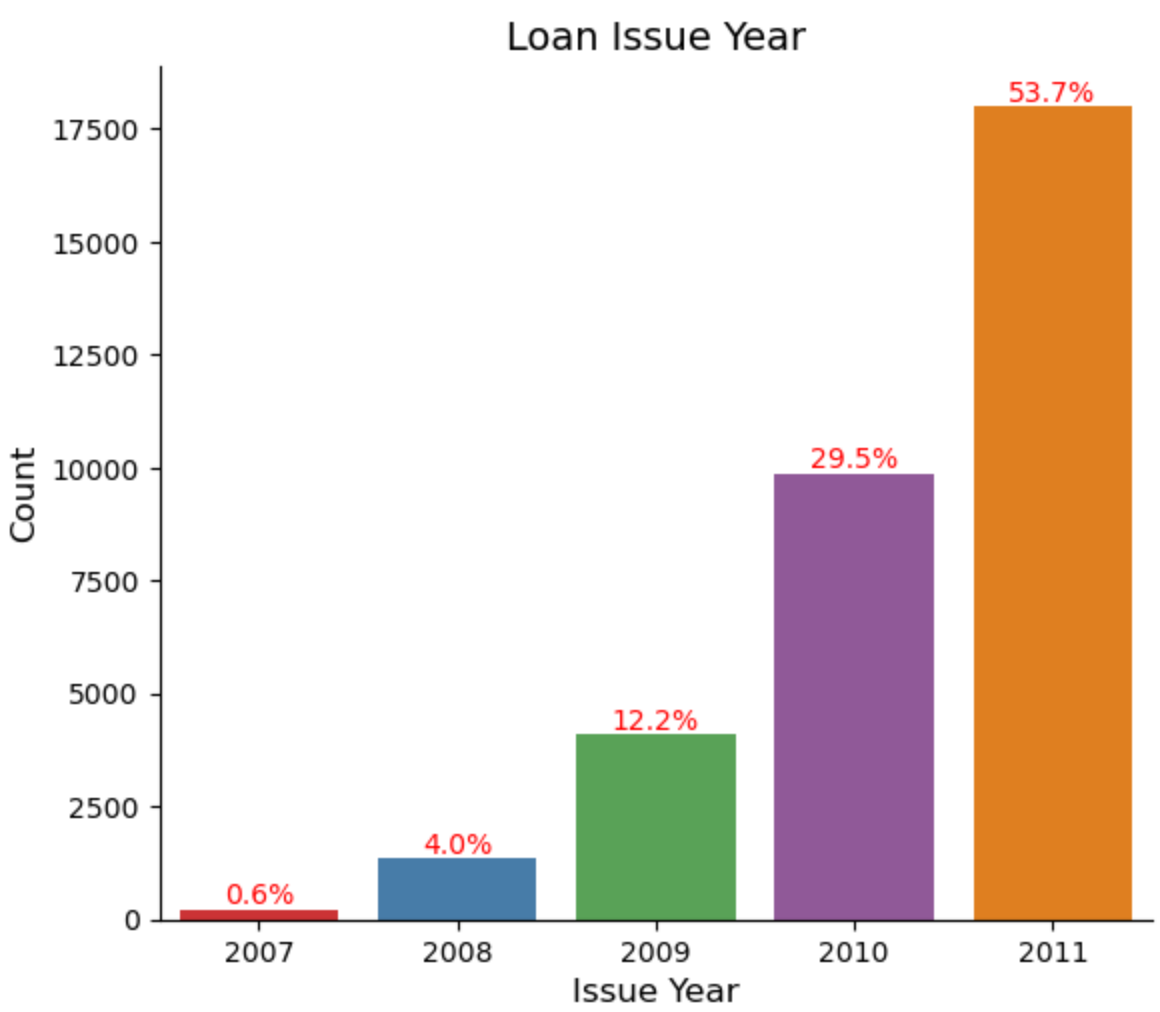
The following are the loan purposes for whcih more than 5% applicants have taken the loan Debt Consilidation 52.4% Credit card 14.2% Home Improvement 8.1% Major purchase 6.2% Small Business 5.0%

In [74]:
```
order_grade = ['A','B','C','D','E','F','G']
plot = sns.catplot(data = loan, x ='grade', kind = 'count', palette = 'Set1', aspect = 1
plt.title('Grade', fontsize = 14)
plt.xlabel('Grade', fontsize =12)
plt.ylabel('Count', fontsize = 12)
ax = plot.facet_axis(0,0)
for p in ax.patches:
     ax.annotate('{:1.1f}%'.format((p.get_height()*100/len(loan))), ((p.get_x()+ p.get_w
                 color = 'red', ha = 'center', va = 'bottom')
```



The above graph shows that most of the applicants fall under category B(30.3%) followed by A and C 25.6%, 20.3% respectively.

In [75]:
```
order_year = ['2007', '2008', '2009', '2010', '2011']
plot = sns.catplot(data = loan , x = 'issue_year', kind = 'count', palette = 'Set1', asp
plt.title('Loan Issue Year', fontsize = 14)
plt.xlabel('Issue Year', fontsize =12)
plt.ylabel('Count', fontsize =12)

ax =plot.facet_axis(0,0)
for p in ax.patches:
    ax.annotate("{:1.1f}%".format((p.get_height()*100/len(loan))), ((p.get_x() + p.get_w
                color = 'red', ha = 'center', va = 'bottom')
```



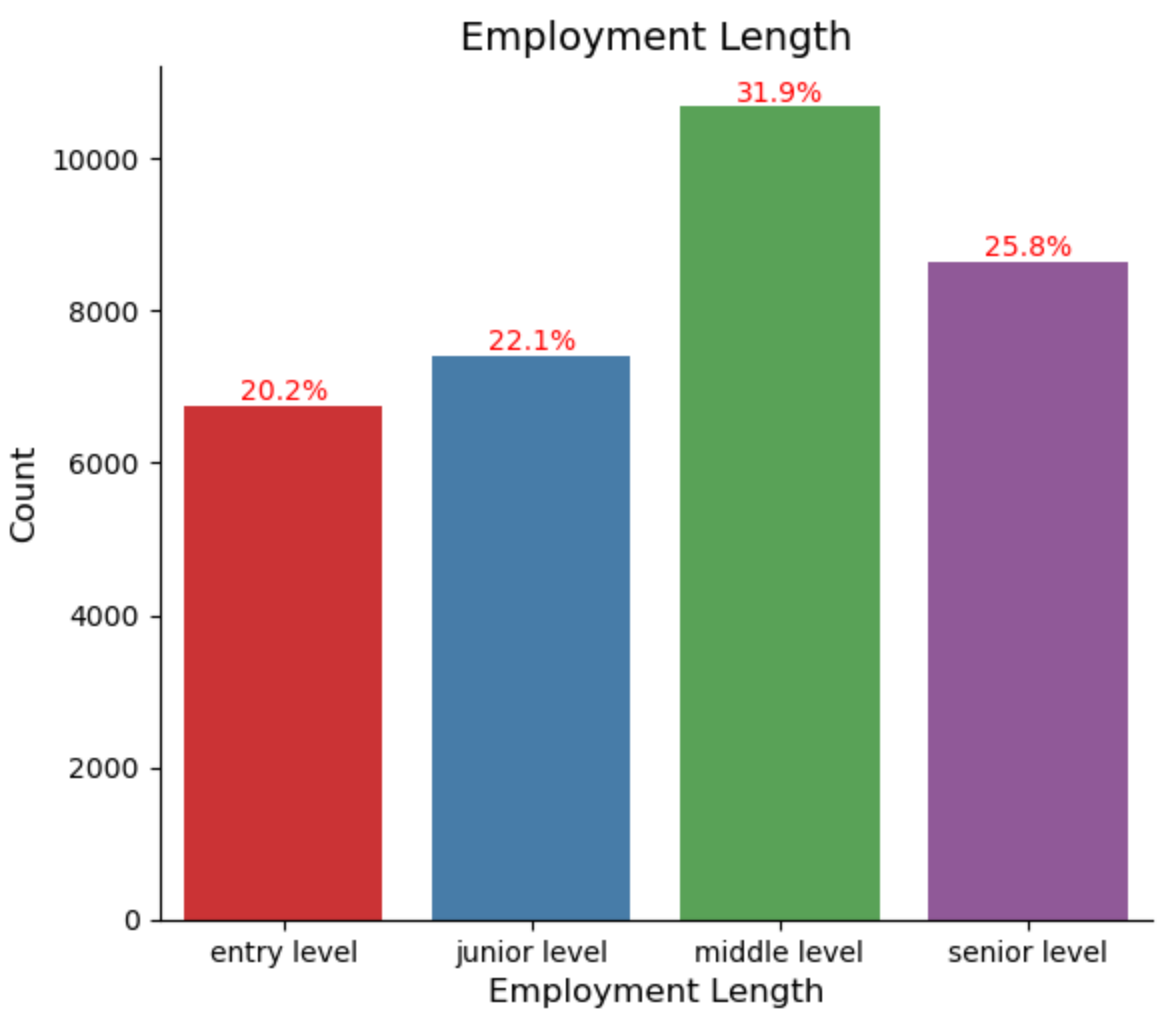
As per the above graph, applicants for loan increased as the year increased, In 2011 the number of applicants for loan was 53.7%, Since the variable issue year does not provide us any direction in the analysis so we will use it for further analysis.

In [76]:
```
loan.categorised_emp_length.value_counts()
```

Out[76]:
```
middle level    10680
senior level     8644
junior level     7410
entry level      6758
Name: categorised_emp_length, dtype: int64
```

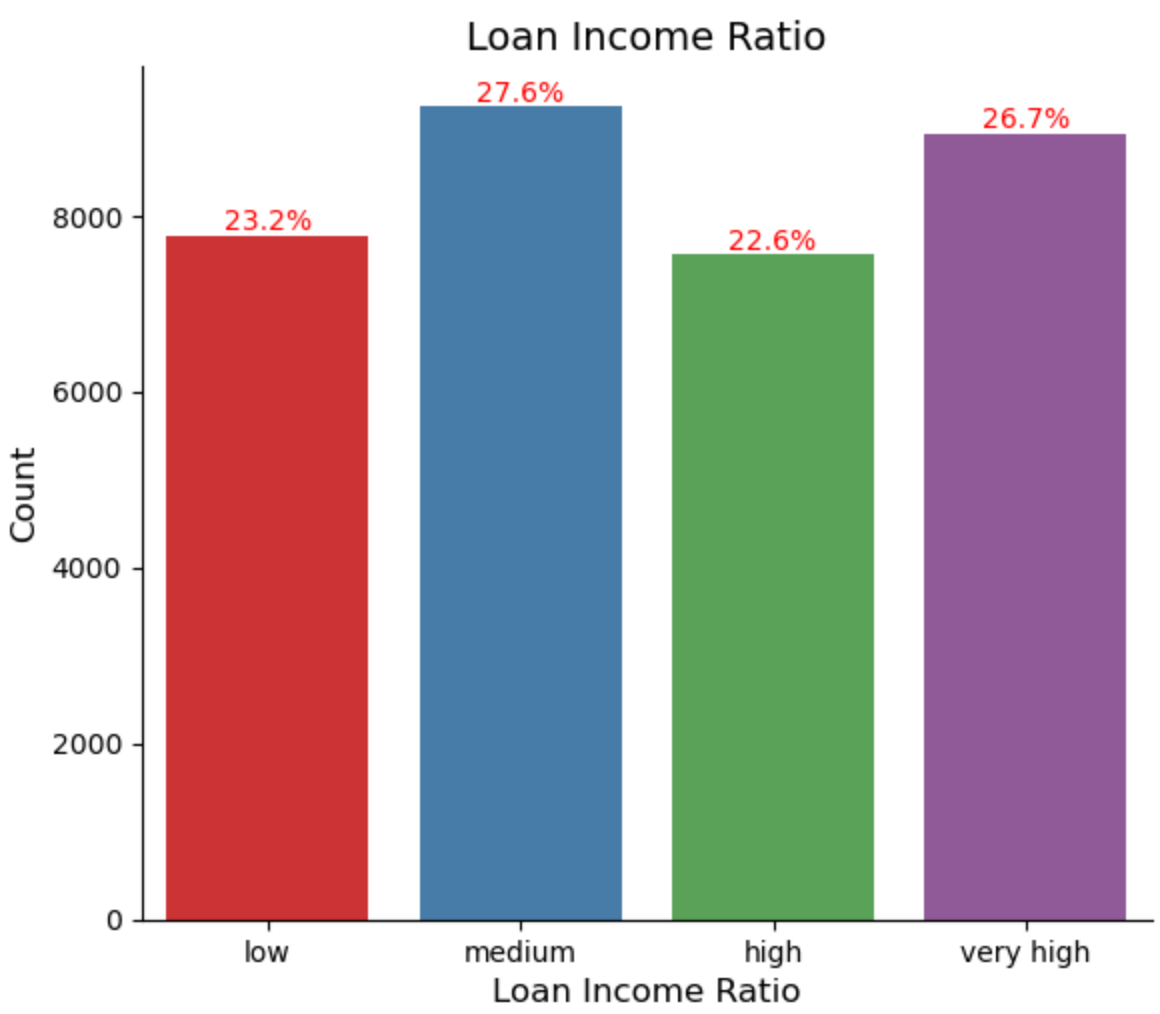
Segmented Univariate analysis

In [96]:
```
ordered = ['entry level', 'junior level', 'middle level', 'senior level']
plot = sns.catplot(data = loan , x = 'categorised_emp_length', kind = 'count', palette =
plt.title('Employment Length', fontsize = 14)
plt.xlabel('Employment Length', fontsize = 12)
plt.ylabel('Count', fontsize = 12)

ax = plot.facet_axis(0,0)

for p in ax.patches:
    ax.annotate("{:1.1f}%".format((p.get_height()*100/len(loan))), ((p.get_x() + p.get_w
                color = 'red', ha = 'center', va = 'bottom')

plt.show()
```



There are more number of loan applicants belonging to the middle level category(31.9%) 4 to 8 Years

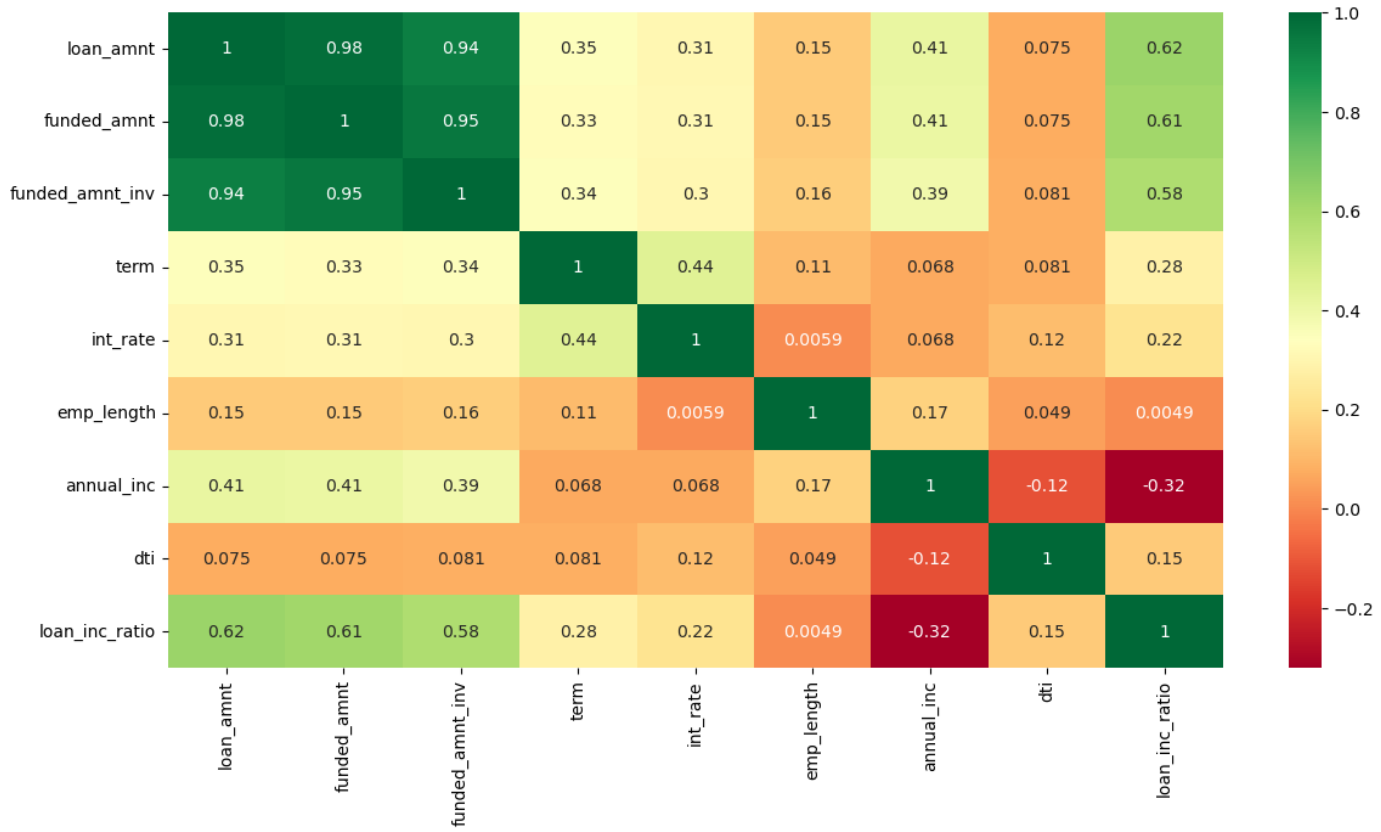
In [97]:
```
ordered = ['low','medium','high', 'very high']
plot = sns.catplot(data = loan, x = 'categorised_loan_inc_ratio', kind = 'count', palett
plt.title('Loan Income Ratio', fontsize = 14)
plt.xlabel('Loan Income Ratio', fontsize = 12)
plt.ylabel('Count', fontsize =12)

ax = plot.facet_axis(0,0)
for p in ax.patches:
    ax.annotate("{:1.1f}%".format((p.get_height()*100/len(loan))), ((p.get_x() + p.get_w
                color = 'red', ha = 'center', va = 'bottom')
```



Bivariate analysis

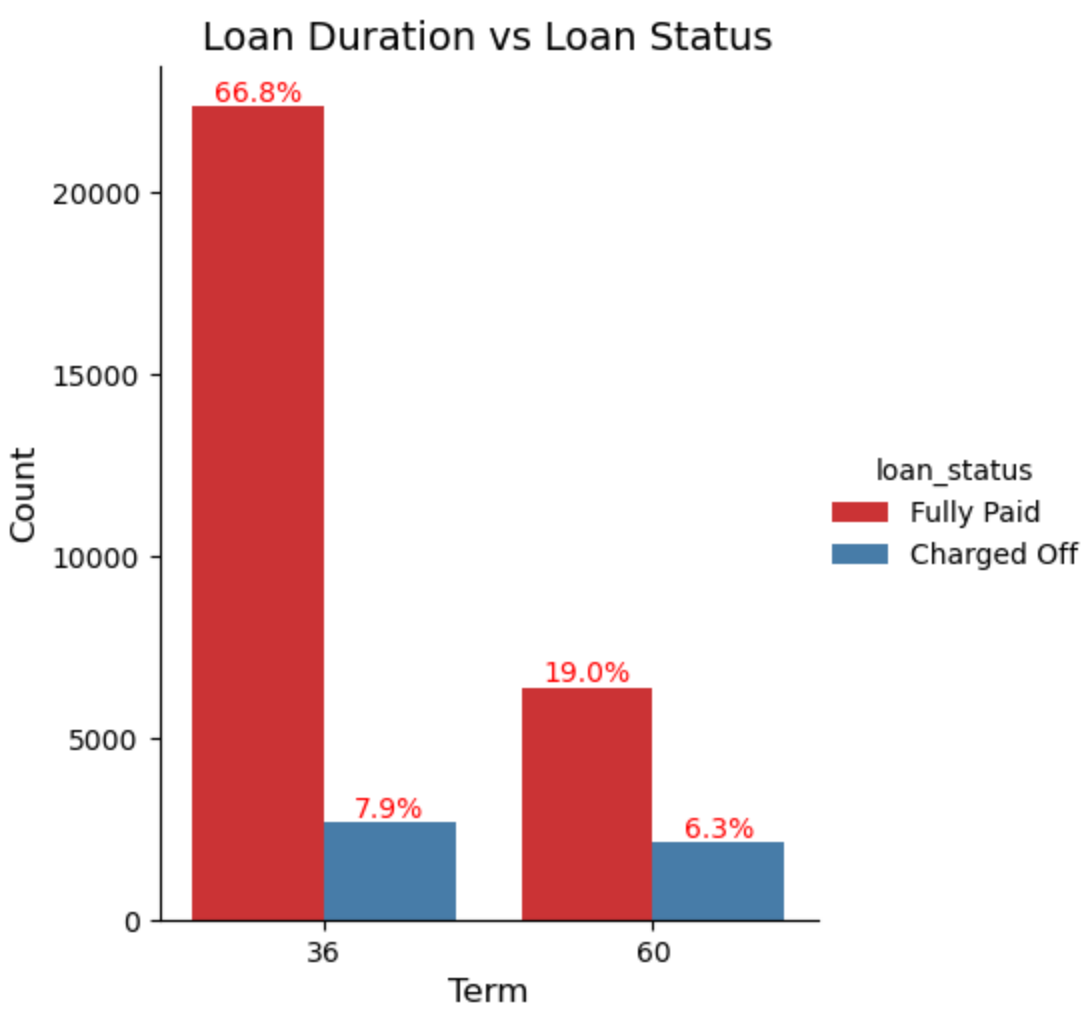
In [98]:
```
loan_correlation = loan.corr()
plt.figure(figsize =(14,7))
sns.heatmap(loan_correlation, annot = True, cmap = 'RdYlGn')
```

Out[98]:
```
<Axes: >
```



The above heatmap shows that the loan amount, funded amount and funded amount investment are very closely correlated, Hence we can take any of them for our analysis.

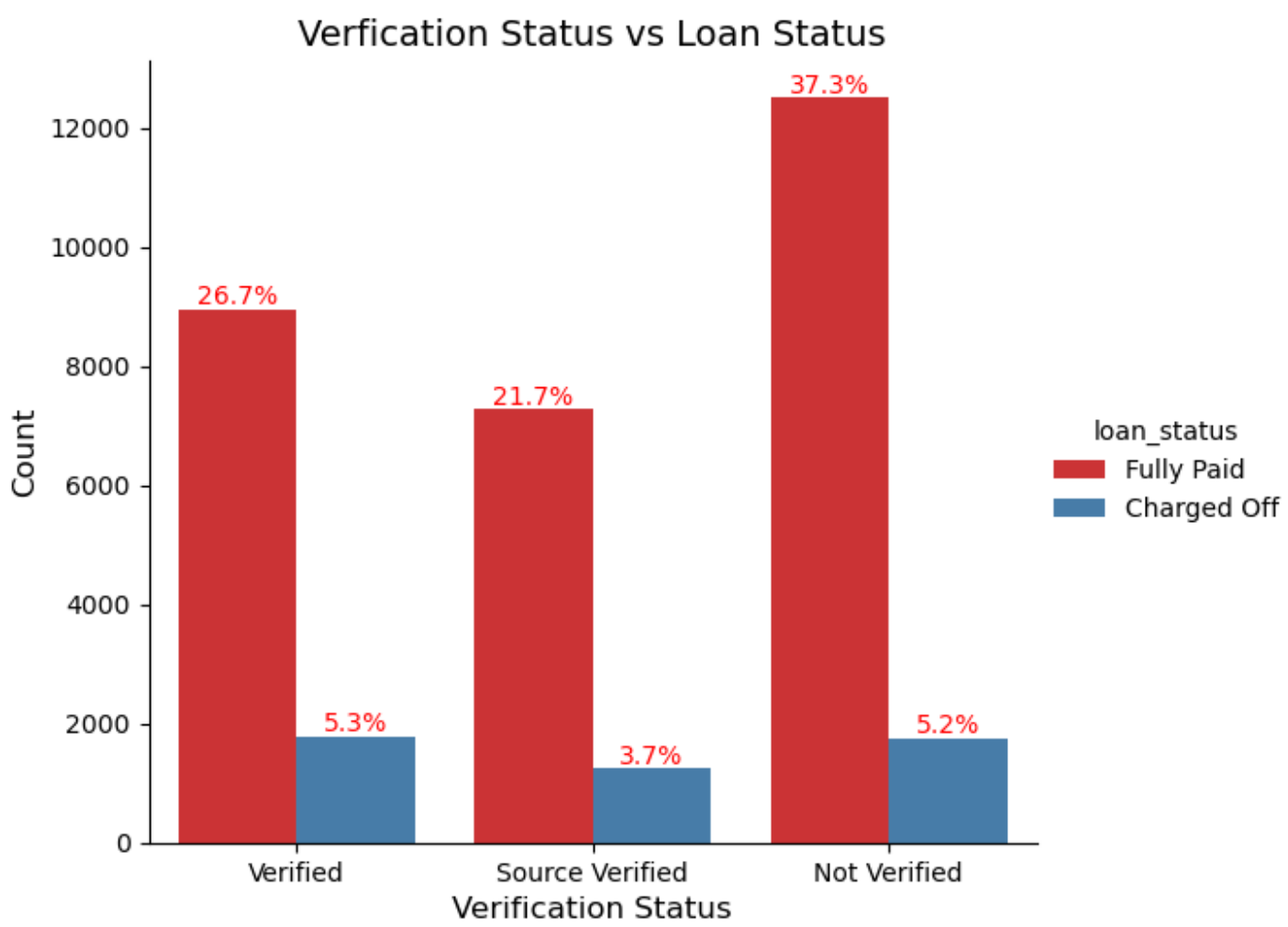
In [99]:
```
#Loan duration vs loan status
plot = sns.catplot(data = loan , x = 'term', hue = 'loan_status', kind = 'count', palett
plt.title('Loan Duration vs Loan Status', fontsize = 14)
plt.xlabel('Term', fontsize = 12)
plt.ylabel('Count', fontsize = 12)

ax = plot.facet_axis(0,0)
for p in ax.patches:
    ax.annotate("{:1.1f}%".format((p.get_height()*100/len(loan))), ((p.get_x()+ p.get_wi
                color = 'red', ha = 'center', va = 'bottom')
```



75% of applicants opted for 36 month loan duration and 66.8% have fully paid while 7.9% are charged off.
On the other hand approx 25% applicants had 60months tenure and 6.3% were charged off.

In [100…

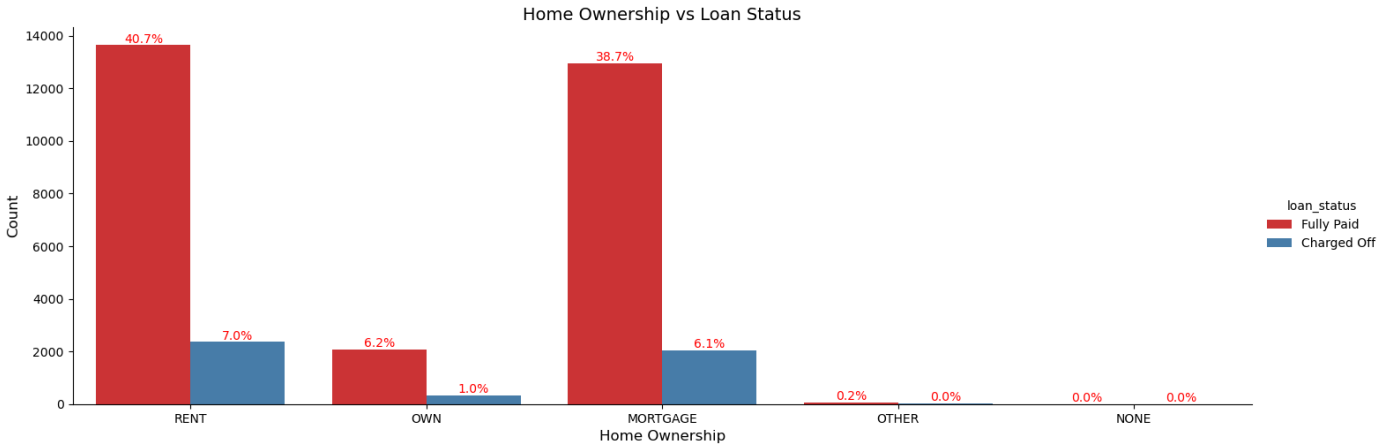
```
#Verification status vs Loan status
plot = sns.catplot(data = loan , x = 'verification_status', kind = 'count', hue = 'loan_
plt.title('Verfication Status vs Loan Status', fontsize = 14)
plt.xlabel('Verification Status', fontsize = 12)
plt.ylabel('Count', fontsize = 12)

ax = plot.facet_axis(0,0)
for p in ax.patches:
    ax.annotate("{:1.1f}%".format((p.get_height()*100/len(loan))), ((p.get_x() + p.get_w
                color = 'red', ha = 'center', va = 'bottom')
```



The above graph shows that the applicants whose income is verified seem to default more, hence we can ignore this as a cause for default for the further analysis.

In [101…

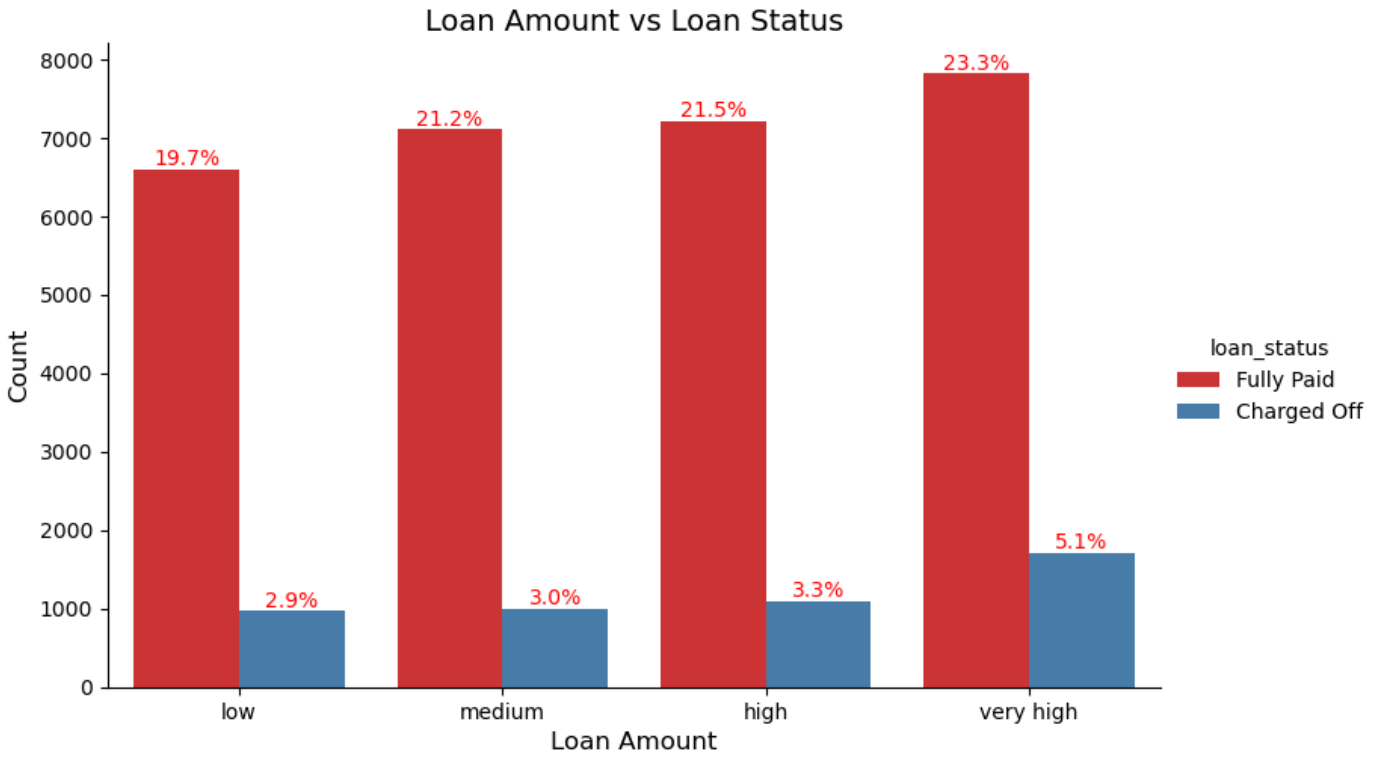
```
#Home_ownership vs Loan_status
plot = sns.catplot(data = loan, x = 'home_ownership', kind = 'count', hue = 'loan_status
plt.title('Home Ownership vs Loan Status', fontsize = 14)
plt.xlabel('Home Ownership', fontsize = 12)
plt.ylabel('Count', fontsize = 12)

ax = plot.facet_axis(0,0)
for p in ax.patches:
    ax.annotate("{:1.1f}%".format((p.get_height()*100/len(loan))), ((p.get_x() + p.get_w
                color ='red', ha= 'center', va = 'bottom')
```



The above graph shows the applicants who are on rent and mortgage are likely to default as compaired to applicants who own homes.

In [102…

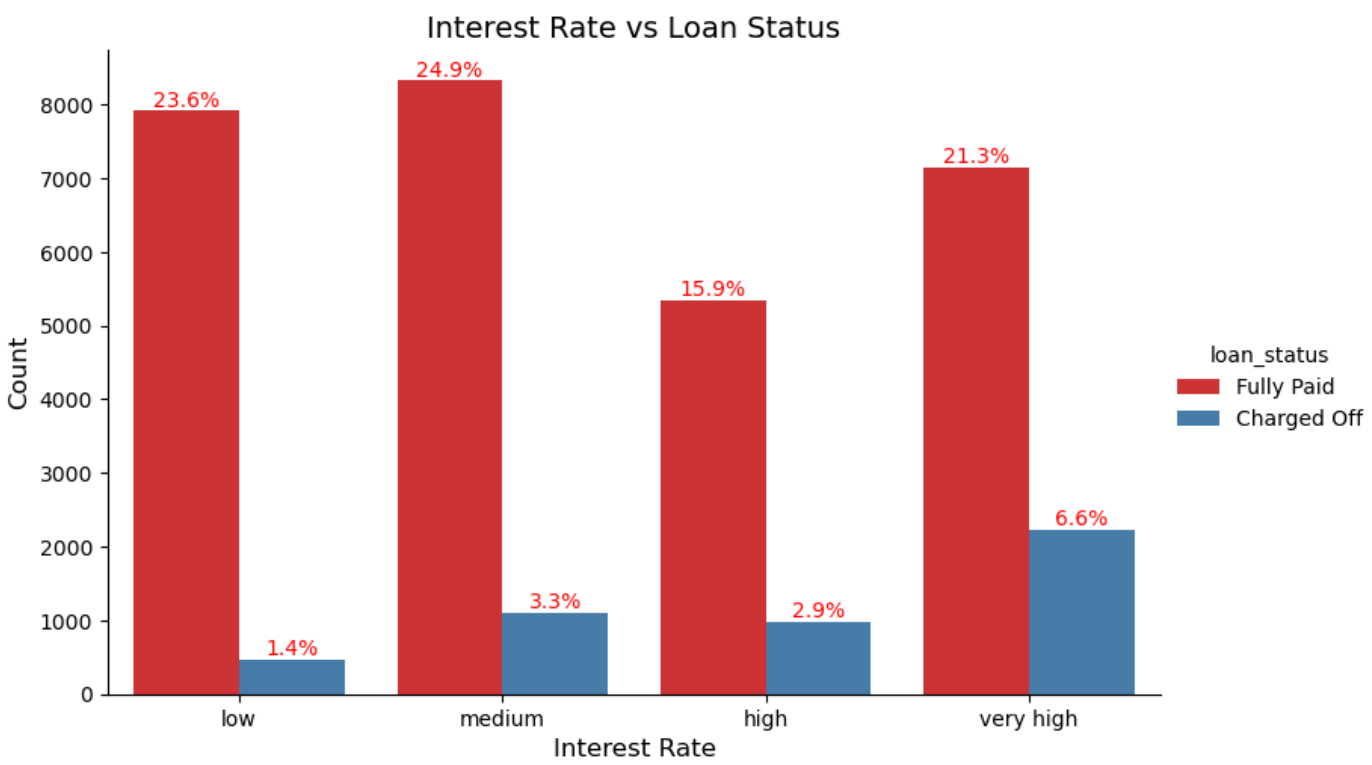
```
#Loan amount vs Loan Status
ordered = ['low', 'medium', 'high', 'very high']
g = sns.catplot(data = loan, x = 'categorised_loan_amnt', kind = 'count', hue = 'loan_st
            order = ordered)
plt.title('Loan Amount vs Loan Status', fontsize = 14)
plt.xlabel('Loan Amount', fontsize = 12)
plt.ylabel('Count', fontsize = 12)

ax = g.facet_axis(0,0)
for p in ax.patches:
    ax.annotate("{:1.1f}%".format((p.get_height()*100/len(loan))), ((p.get_x() + p.get_w
                color = 'red', ha = 'center', va = 'bottom')
plt.show()
```



The above graph shows the applicants having high loan amount are likely to default.

In [103…

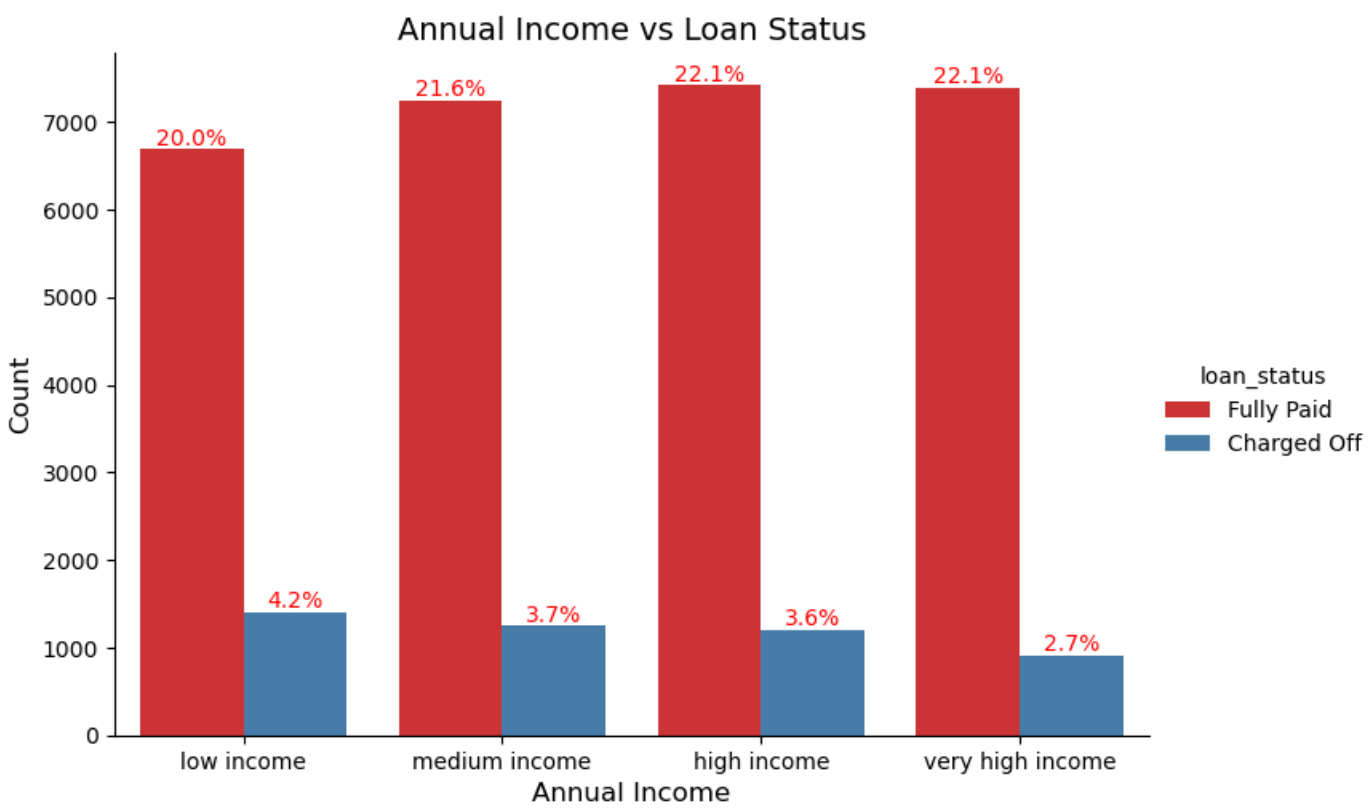
```
#Interest rate vs Loan status
ordered = ['low', 'medium', 'high', 'very high']
g = sns.catplot(data = loan, x = 'categorised_int_rate_perc', kind = 'count', hue = 'loa
            order = ordered)
plt.title('Interest Rate vs Loan Status', fontsize = 14)
plt.xlabel('Interest Rate', fontsize = 12)
plt.ylabel('Count', fontsize = 12)

ax = g.facet_axis(0,0)
for p in ax.patches:
    ax.annotate("{:1.1f}%".format((p.get_height()*100/len(loan))), ((p.get_x() + p.get_w
                color = 'red', ha = 'center', va = 'bottom')
plt.show()
```



The above graph shows the applicants having higher interest rates are likely to default.

In [104…

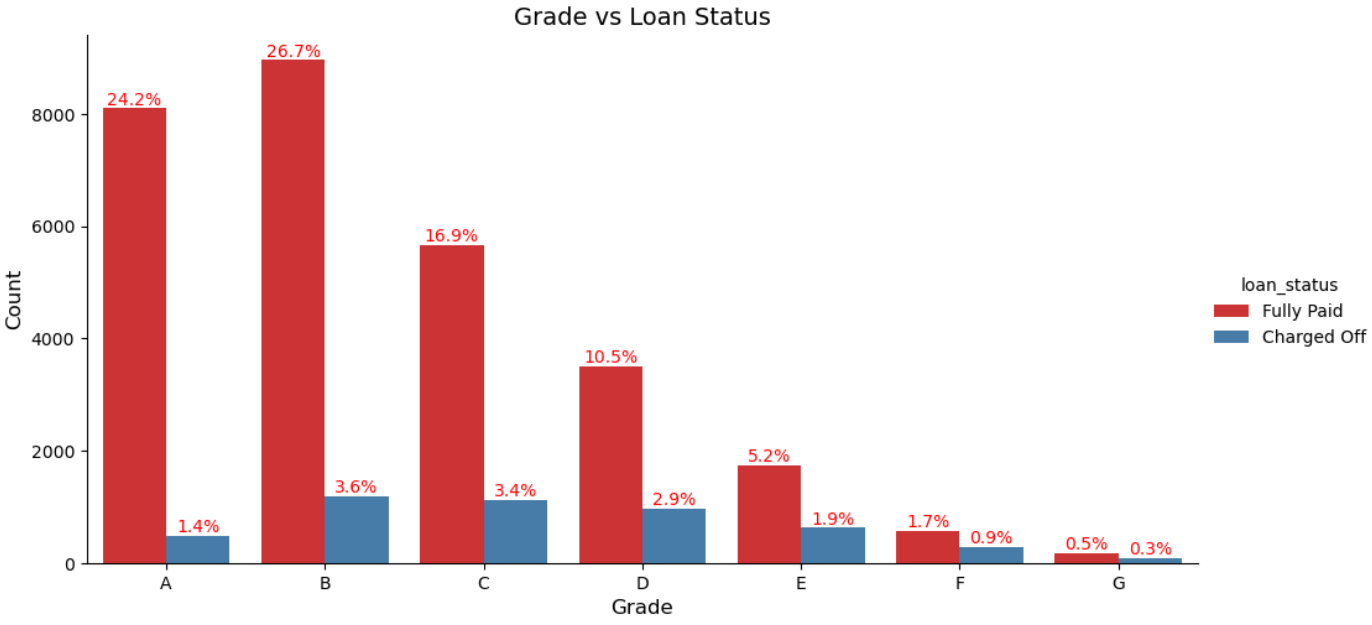
```
#Annual Income vs Loan Status
ordered = ['low income','medium income', 'high income', 'very high income']
g= sns.catplot(data = loan, x = 'categorised_annual_inc', kind = 'count', hue = 'loan_st
           order = ordered)
plt.title('Annual Income vs Loan Status', fontsize = 14)
plt.xlabel('Annual Income', fontsize = 12)
plt.ylabel('Count', fontsize = 12)

ax = g.facet_axis(0,0)
for p in ax.patches:
    ax.annotate("{:1.1f}%".format((p.get_height()*100/len(loan))), ((p.get_x() + p.get_w
                color = 'red', ha = 'center', va = 'bottom')
```



The above graph shows the applicants having higher annual income are less likely to default.

In [105…

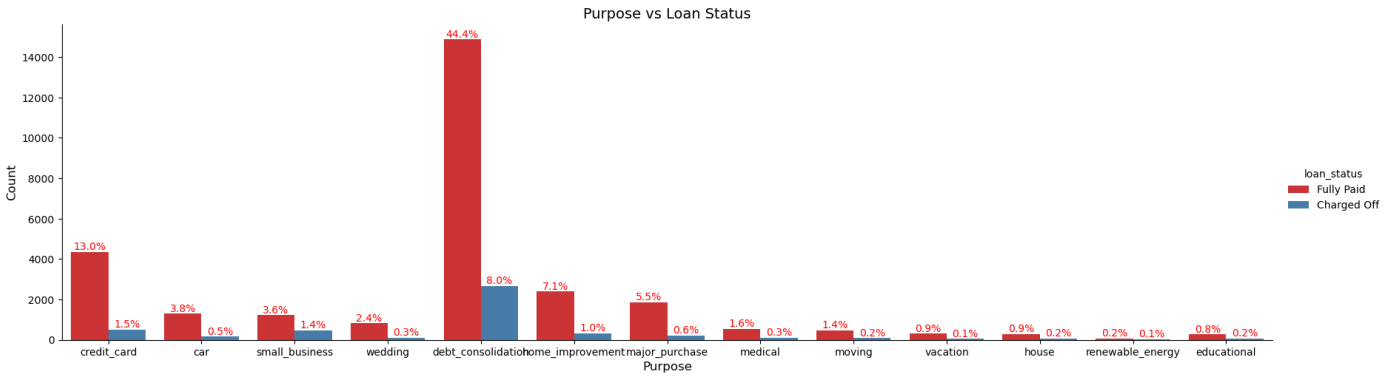
```
#Grade vs Loan Status
ordered = ['A', 'B', 'C', 'D', 'E', 'F', 'G']
n = sns.catplot(data = loan , x = 'grade', kind = 'count', hue = 'loan_status', palette
plt.title('Grade vs Loan Status', fontsize = 14)
plt.xlabel('Grade', fontsize = 12)
plt.ylabel('Count', fontsize = 12)


ax = n.facet_axis(0,0)
for p in ax.patches:
    ax.annotate("{:1.1f}%".format((p.get_height()*100/len(loan))), ((p.get_x()+p.get_wid
                color = 'red', ha = 'center', va = 'bottom')
```



In [106…

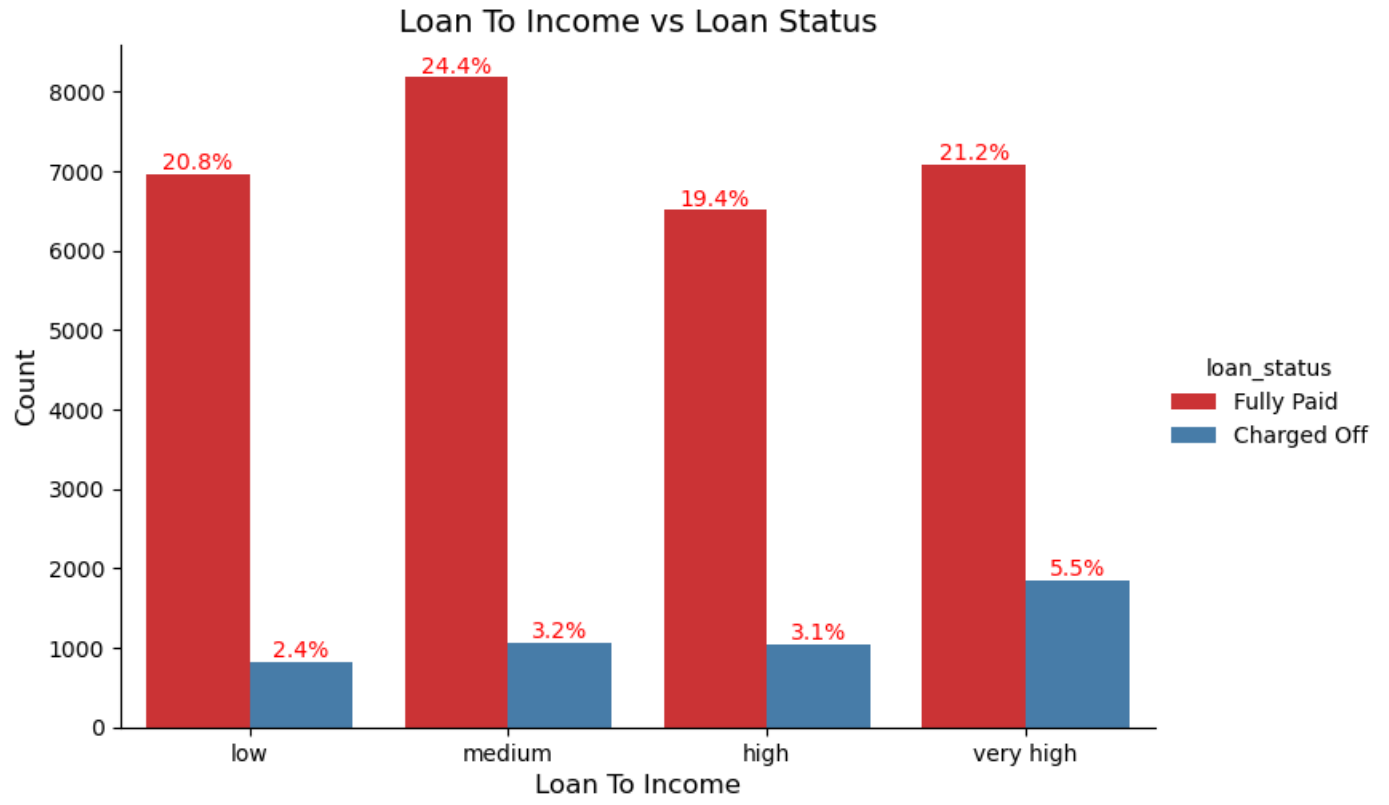
```
#Purpose vs Loan Status
atplot(data = loan , x = 'purpose', kind = 'count', hue = 'loan_status', pal
```

```
plt.title('Purpose vs Loan Status', fontsize = 14)
plt.xlabel('Purpose', fontsize = 12)
plt.ylabel('Count', fontsize = 12)

ax = plot.facet_axis(0,0)
for p in ax.patches:
    ax.annotate("{:1.1f}%".format((p.get_height()*100/len(loan))), ((p.get_x() + p.get_w
                color = 'red', ha = 'center', va= 'bottom')

plt.show()
```



In [107…

```
#Loan to income vs Loan Status
ordered = ['low', 'medium', 'high', 'very high']
g = sns.catplot(data = loan, x = 'categorised_loan_inc_ratio', kind = 'count', hue = 'lo
           order = ordered)
plt.title('Loan To Income vs Loan Status', fontsize = 14)
plt.xlabel('Loan To Income', fontsize = 12)
plt.ylabel('Count', fontsize = 12)

ax = g.facet_axis(0,0)
for p in ax.patches:
    ax.annotate("{:1.1f}%".format((p.get_height()*100/len(loan))), ((p.get_x()+ p.get_wi
                color = 'red', ha = 'center', va = 'bottom')
plt.show()
```



h shows that the applicants having very high loan to income ratio are likely to default

Bivariate Analysis outcome

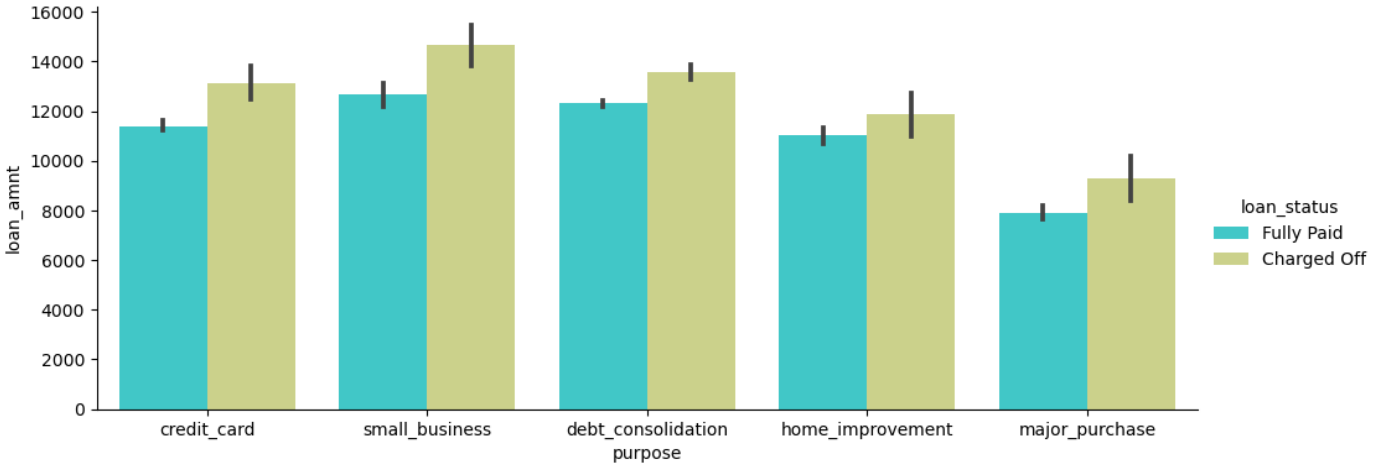
Following are the type of applicants who are likely to default:

- Applicants with a loan term of 36 months(7.4%)
- Applicants who stay in rented houses and followed by applicants who stay at mortgage (7.0%) and (6.1%) respectively
- Applicants who have taken very high loan amount (>= 15000) (5.1%)
- Applicants paying very (> = 14%) interest rates default more (6.6%)
- Applicants with low annual income (41000) (4.2%)
- Applicants identified as grade B tend to default more (3.6%) followed by C , D (3.4%) and (2.9%) respectively
- Applicants with high income to debt ratio (between 13 and 18 inclusive) (4.5%)
- Applicants who have very high loan to income ratio (>=25) (5.5%)
- The default rate in terms of purpose of loan is as follows: Debt Consolidation 8.0% Credit Card 1.5% Small Business 1.4% Home Improvement 1.0% Major Purchase 0.6%

Multivariate Analysis: Analysing more than 2 columns

In [108…

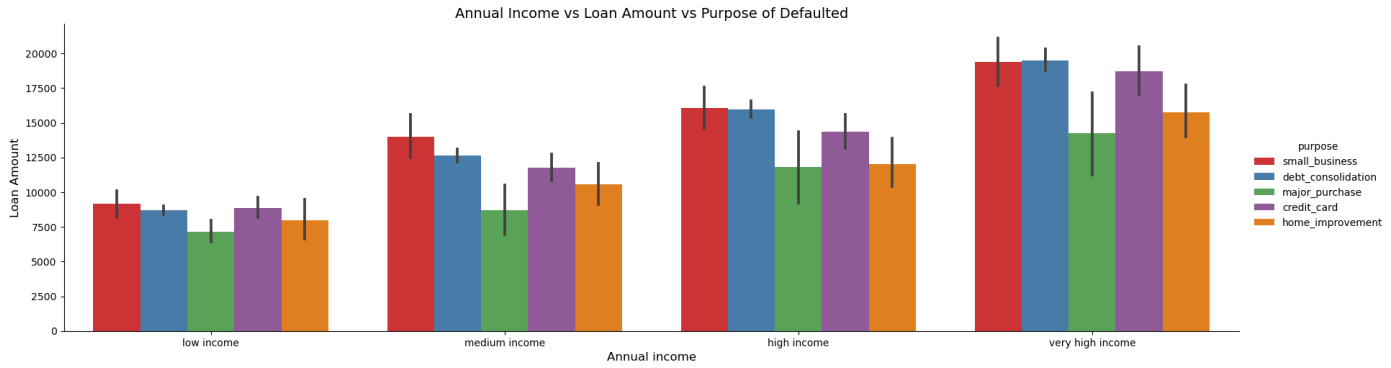
```
#Loan status vs Purpose vs Loan amount

selected_purposes = ['debt_consolidation', 'credit_card', 'small_business', 'home_improv
filtered_df = loan[loan['purpose'].isin(selected_purposes)]

sns.catplot(y = 'loan_amnt', x = 'purpose', hue ='loan_status', data = filtered_df, kind
            ,height = 4)
plt.show()
```



In [109…

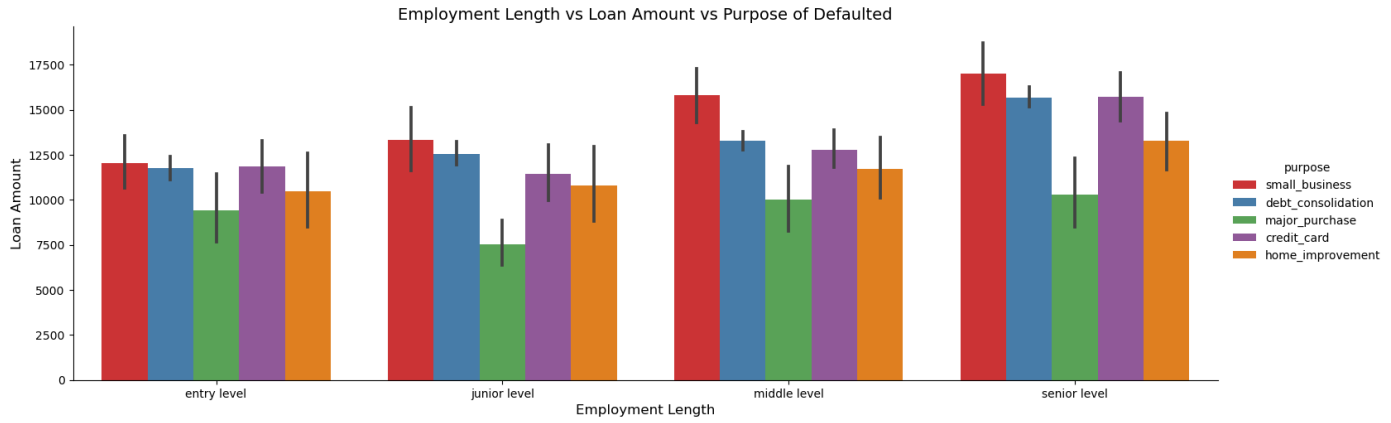
```
#Annual income vs Loan amount vs purpose for charged off
df = filtered_df[filtered_df['loan_status'] == 'Charged Off']
ordered = ['low income', 'medium income', 'high income', 'very high income']
sns.catplot(data = df, x = 'categorised_annual_inc', y = 'loan_amnt', hue = 'purpose', a
            palette = 'Set1', order = ordered, kind = 'bar')
plt.title('Annual Income vs Loan Amount vs Purpose of Defaulted',fontsize = 14)
plt.xlabel('Annual income',fontsize = 12)
```

```
plt.ylabel('Loan Amount', fontsize = 12)
plt.show()
```



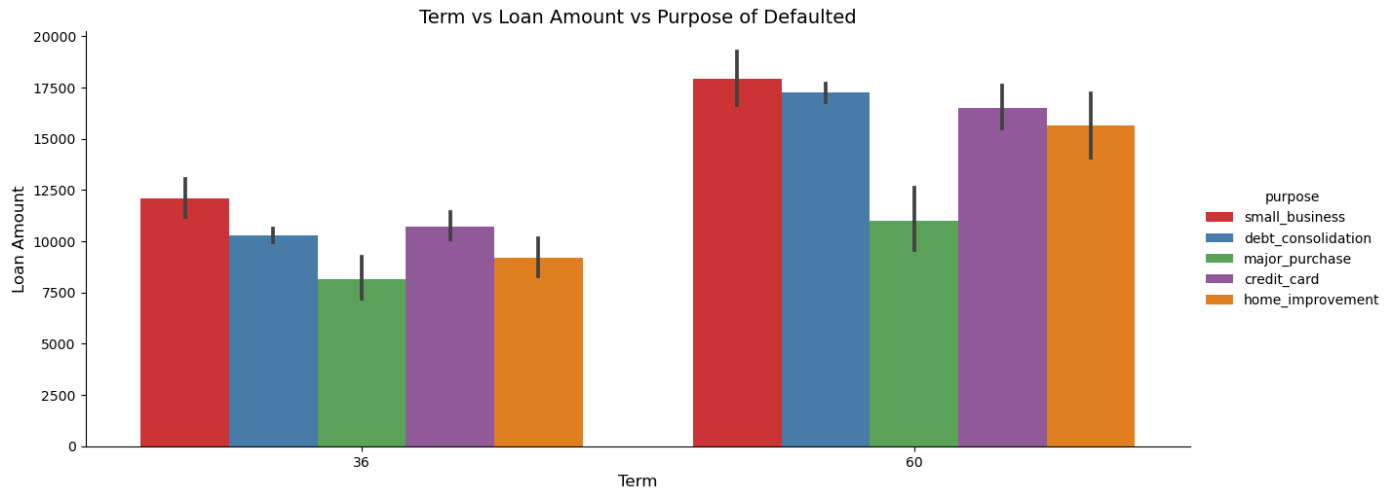
In [110…

```
#Employment length vs loan Amount vs Purpose for charged off loan status
ordered = ['entry level', 'junior level', 'middle level', 'senior level']
sns.catplot(data = df, x = 'categorised_emp_length', y = 'loan_amnt', hue = 'purpose', a
            palette = 'Set1', kind = 'bar', order = ordered)
plt.title('Employment Length vs Loan Amount vs Purpose of Defaulted',fontsize = 14)
plt.xlabel('Employment Length',fontsize = 12)
plt.ylabel('Loan Amount', fontsize = 12)
plt.show()
```



In [111…

```
#Term vs loan Amount vs Purpose for charged off loan status
sns.catplot(data = df, x = 'term', y = 'loan_amnt', hue = 'purpose', aspect = 2.5,
            palette = 'Set1', kind = 'bar')
plt.title('Term vs Loan Amount vs Purpose of Defaulted',fontsize = 14)
plt.xlabel('Term',fontsize = 12)
plt.ylabel('Loan Amount', fontsize = 12)
plt.show()
```



Multivariate analysis outcome:

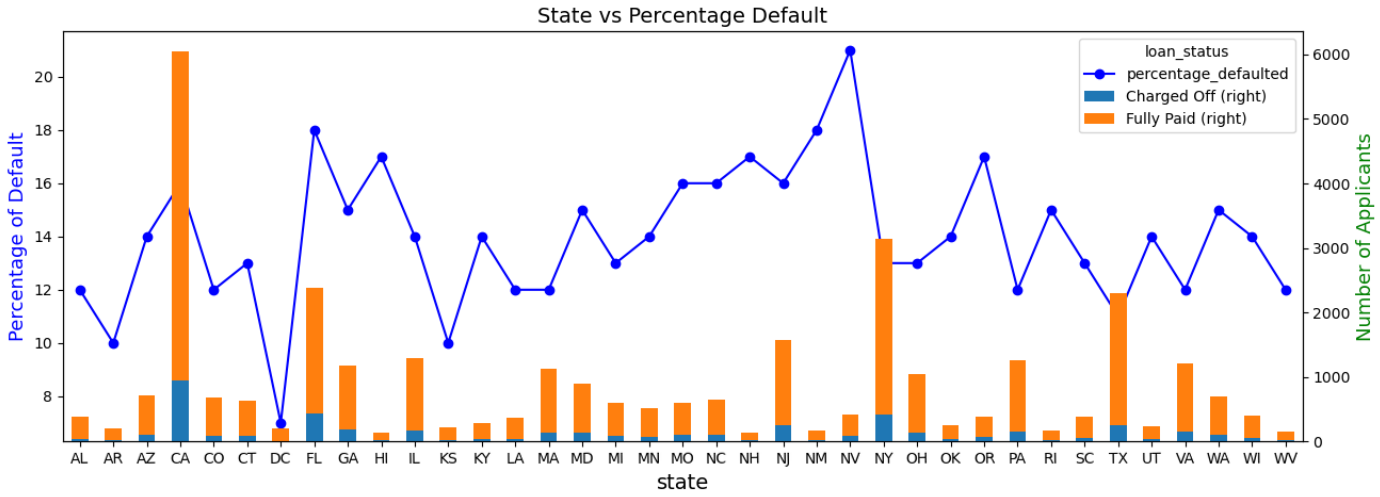
Following are the loan applicants who are likely to default: Applicants who take loan for small business. Applicants whose annual income is in the category of low and medium have defaulted moreon small business, while as the applicants who are categorised in high and very have defaulted in small business and debt consolidation. Irrespective of employment length, those who have taken loan for small business have defaulted. Applicants with a 60 months term duration for small business.

- Hence we can infer that default rate is more for loan taken on small business followed by debt consolidation.

In [112…

```
#Stack Bar chart and Line Chart

def change_width(ax, new_value):
    for patch in ax.patches:
        patch.set_width(new_value)
        patch.set_x(patch.get_x() + (patch.get_width() - new_value) * 0.5)

def plot_map(crosstab, partialtitle, label_name, value):
    # Extract relevant columns from crosstab
    lineplot = crosstab[['percentage_defaulted']]
    barPlot = crosstab[['Charged Off', 'Fully Paid']]

    # Plot line and bar charts on the same axes
    ax = lineplot.plot(figsize=(20, 8), marker='o', color='b')
    ax3 = barPlot.plot(kind='bar', ax=ax, rot=1, secondary_y=True, stacked=True, figsize

    # Set plot title and labels
    ax.set_title(f'{partialtitle} vs Percentage Default', fontsize=14)
    ax.set_xlabel(label_name, fontsize=14)
    ax.set_ylabel('Percentage of Default', color='b', fontsize=13)
    ax3.set_ylabel('Number of Applicants', color='g', fontsize=13)

    # Adjust bar width using the change_width function
    change_width(ax3, value)
    plt.show()

# Sample code for testing
filter_states = loan['addr_state'].value_counts()
filter_states = filter_states[filter_states < 100]
filter_states_df = loan[~loan['addr_state'].isin(filter_states.index)]

filter_states_crosstab = pd.crosstab(filter_states_df['addr_state'], filter_states_df['l
filter_states_crosstab.drop(filter_states_crosstab.tail(1).index, inplace=True)
filter_states_crosstab['percentage_defaulted'] = round(100 * (filter_states_crosstab['Ch

# Display crosstab
display(filter_states_crosstab)

# Plot the map
plot_map(filter_states_crosstab, 'State', 'state', 0.50)
```

loan_status addr_state	Charged Off	Fully Paid	All	percentage_defaulted
AL	46	338	384	12.0
AR	21	187	208	10.0
AZ	99	616	715	14.0
CA	951	5103	6054	16.0
CO	80	600	680	12.0
CT	82	550	632	13.0
DC	13	184	197	7.0
FL	433	1958	2391	18.0
GA	179	1003	1182	15.0
HI	24	120	144	17.0
IL	178	1123	1301	14.0
KS	22	199	221	10.0
KY	39	246	285	14.0
LA	45	327	372	12.0
MA	133	994	1127	12.0
MD	138	754	892	15.0
MI	80	515	595	13.0
MN	72	451	523	14.0
MO	98	499	597	16.0
NC	103	547	650	16.0
NH	23	116	139	17.0
NJ	252	1323	1575	16.0
NM	29	135	164	18.0
NV	87	323	410	21.0
NY	417	2724	3141	13.0
OH	131	908	1039	13.0
OK	35	213	248	14.0
OR	66	322	388	17.0
PA	149	1109	1258	12.0
RI	24	140	164	15.0
SC	48	330	378	13.0
TX	256	2042	2298	11.0
UT	33	196	229	14.0
VA	147	1063	1210	12.0
WA	104	598	702	15.0
WI	54	341	395	14.0
WV	18	134	152	12.0



In [113…
```
loan.categorised_int_rate_perc.value_counts()
```

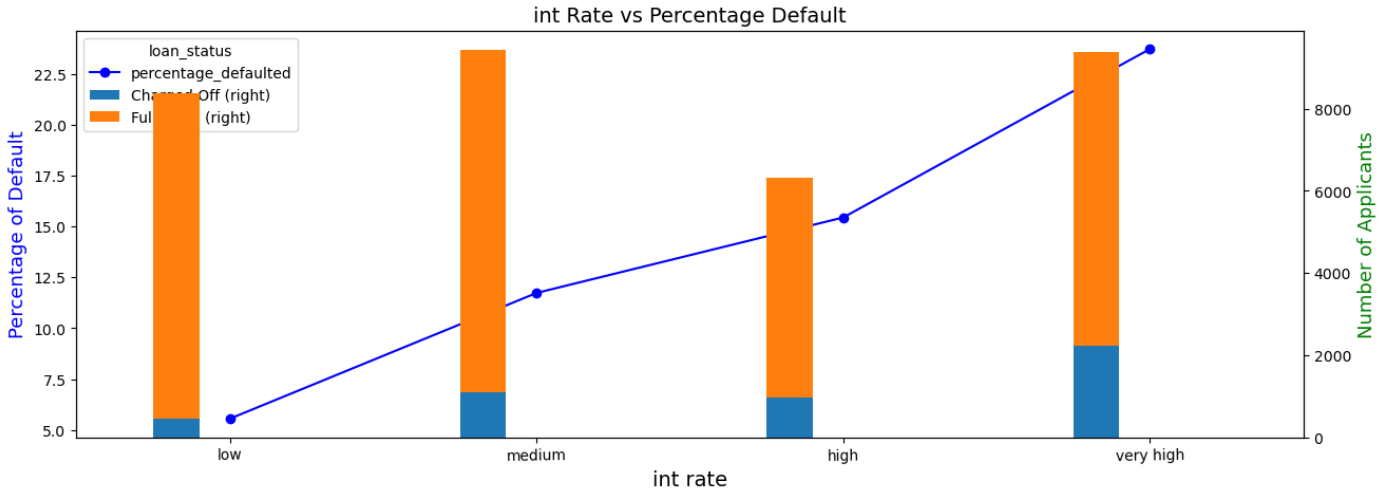
Out[113]:
```
medium       9430
very high    9366
low          8382
high         6314
Name: categorised_int_rate_perc, dtype: int64
```

In [114…
```
loan.categorised_annual_inc.isnull().sum()
```

Out[114]:
```
0
```

In [115…
```
# Define the order of categories
order_category = ['low', 'medium', 'high', 'very high']

# Create a copy of the DataFrame and categorize the column
int_rate_loan_df = loan.copy()
int_rate_loan_df['categorised_int_rate_perc'] = pd.Categorical(int_rate_loan_df['categor

# Sort DataFrame by the categorized column
int_rate_loan_df = int_rate_loan_df.sort_values('categorised_int_rate_perc')

# Create a cross-tabulation
int_rate_crosstab = pd.crosstab(int_rate_loan_df['categorised_int_rate_perc'], int_rate_
int_rate_crosstab.drop(int_rate_crosstab.tail(1).index, inplace=True)

# Calculate the percentage defaulted
int_rate_crosstab['percentage_defaulted'] = round(100 * (int_rate_crosstab['Charged Off'

# Display the cross-tabulation
display(int_rate_crosstab)

# Plot the map
plot_map(int_rate_crosstab, 'int Rate', 'int rate', .15)
```

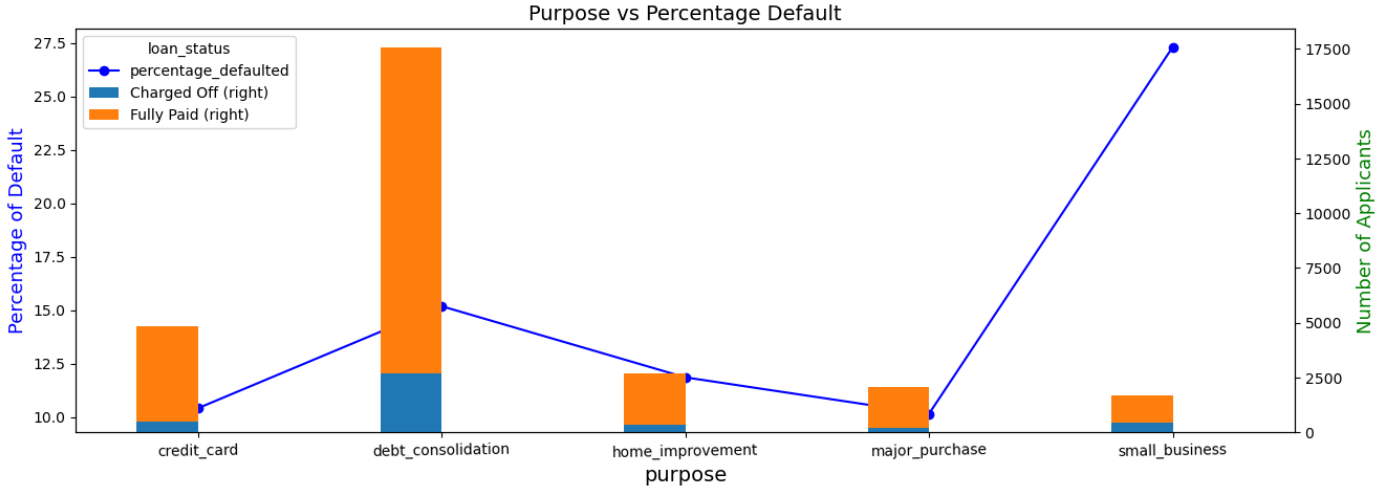
loan_status	Charged Off	Fully Paid	All	percentage_defaulted
categorised_int_rate_perc				
low	465	7917	8382	5.55
medium	1106	8324	9430	11.73
high	975	5339	6314	15.44
very high	2222	7144	9366	23.72



In [116…

```
# filter_df  -- percentage default for purpose

#create a crosstab
purposecrosstab = pd.crosstab(filtered_df['purpose'],filtered_df['loan_status'], margins
purposecrosstab.drop(purposecrosstab.tail(1).index, inplace = True)
purposecrosstab['percentage_defaulted'] = round(100*((purposecrosstab['Charged Off']/pur
#Display crosstab
display(purposecrosstab)
#plot the map
plot_map(purposecrosstab, 'Purpose', 'purpose', .25)
```

loan_status	Charged Off	Fully Paid	All	percentage_defaulted
purpose				
credit_card	506	4356	4862	10.41
debt_consolidation	2665	14875	17540	15.19
home_improvement	321	2386	2707	11.86
major_purchase	209	1851	2060	10.15
small_business	459	1222	1681	27.31



Applicants who have taken loan for small business (27.31) tend to default more

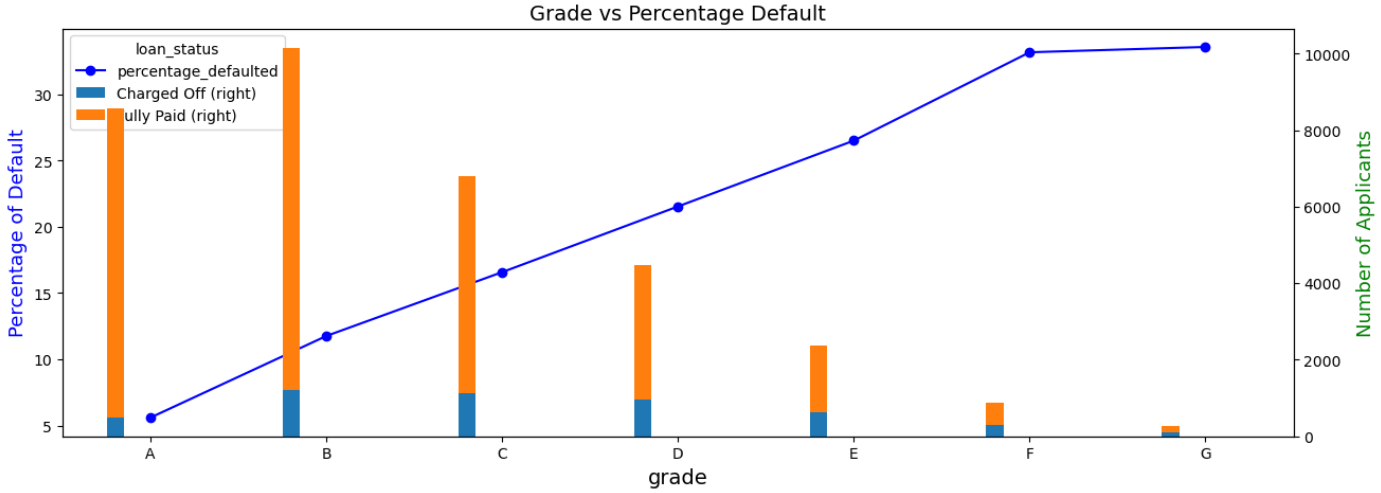
In [117…

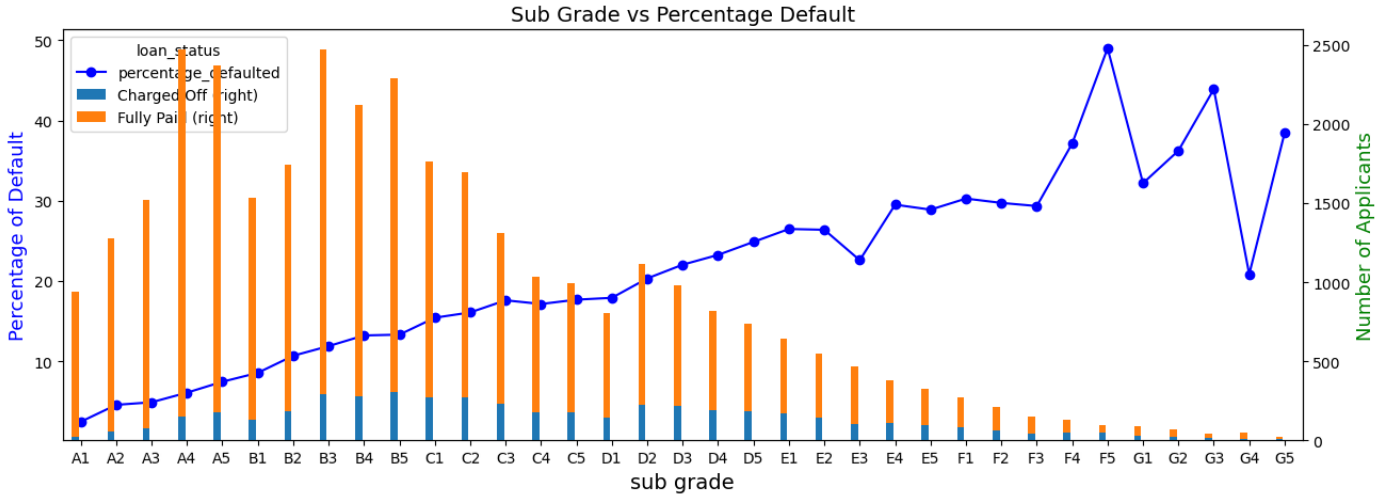
```
#Percentage default for Grade

#Sort Grade based on custom sorting
grade_df = loan
grade_df.sort_values(['grade'])
```

```
sub_grade_df = loan
sub_grade_df.sort_values(['sub_grade'])

#create a cross tab function

grade_crosstab = pd.crosstab(grade_df['grade'], grade_df['loan_status'], margins = True)
grade_crosstab.drop(grade_crosstab.tail(1).index, inplace = True)
grade_crosstab['percentage_defaulted'] = round(100*((grade_crosstab['Charged Off']/ grad
display(grade_crosstab)

sub_grade_crosstab = pd.crosstab(sub_grade_df['sub_grade'],sub_grade_df['loan_status'],
sub_grade_crosstab.drop(sub_grade_crosstab.tail(1).index, inplace = True)
sub_grade_crosstab['percentage_defaulted'] = round(100*((sub_grade_crosstab['Charged Off
display(sub_grade_crosstab)

plot_map(grade_crosstab, 'Grade', 'grade', .10)
plot_map(sub_grade_crosstab, 'Sub Grade', 'sub grade', .20)
```

loan_status	Charged Off	Fully Paid	All	percentage_defaulted
grade				
A	481	8101	8582	5.60
B	1194	8959	10153	11.76
C	1126	5665	6791	16.58
D	961	3500	4461	21.54
E	628	1741	2369	26.51
F	287	578	865	33.18
G	91	180	271	33.58

loan_status sub_grade	Charged Off	Fully Paid	All	percentage_defaulted
A1	23	919	942	2.44
A2	58	1218	1276	4.55
A3	74	1447	1521	4.87
A4	150	2323	2473	6.07
A5	176	2194	2370	7.43
B1	131	1403	1534	8.54
B2	186	1558	1744	10.67
B3	293	2175	2468	11.87
B4	280	1841	2121	13.20
B5	304	1982	2286	13.30
C1	272	1493	1765	15.41
C2	272	1421	1693	16.07
C3	230	1078	1308	17.58
C4	177	857	1034	17.12
C5	175	816	991	17.66
D1	144	661	805	17.89
D2	226	887	1113	20.31
D3	216	765	981	22.02
D4	191	631	822	23.24
D5	184	556	740	24.86
E1	171	475	646	26.47
E2	144	402	546	26.37
E3	105	360	465	22.58
E4	113	270	383	29.50
E5	95	234	329	28.88
F1	82	189	271	30.26
F2	63	149	212	29.72
F3	44	106	150	29.33
F4	49	83	132	37.12
F5	49	51	100	49.00
G1	28	59	87	32.18
G2	25	44	69	36.23
G3	18	23	41	43.90
G4	10	38	48	20.83
G5	10	16	26	38.46



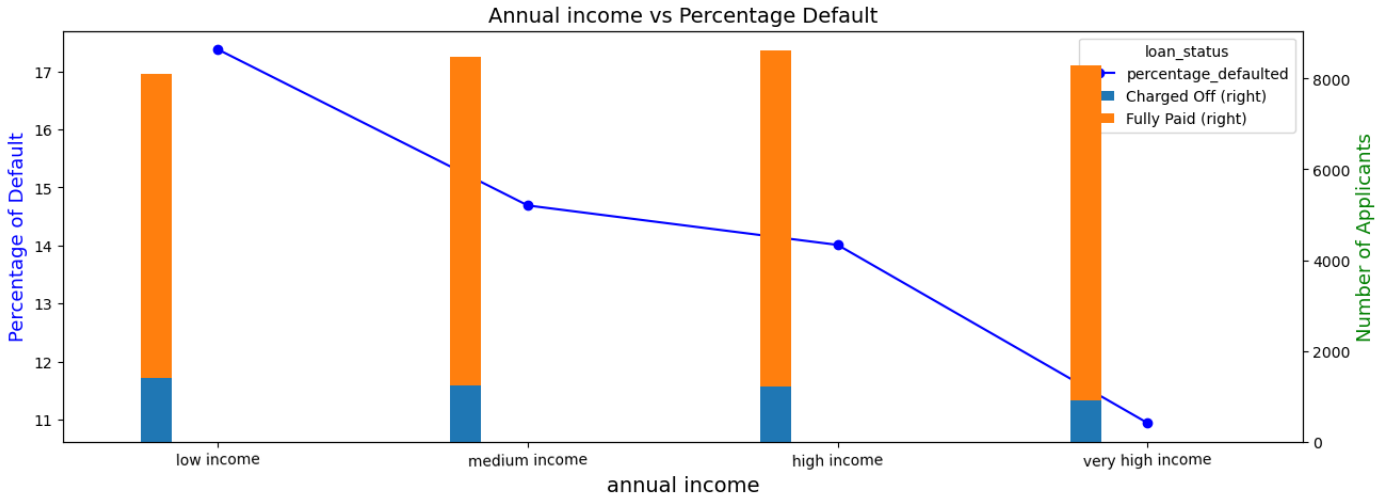


From the above graph it can be observed that probability of applicants increases with the grades from A to G

In [118…

```
#Percentage default for annual income

#sort categorised annual income on custom sorting
order_category = ['low income', 'medium income', 'high income', 'very high income']
annual_inc_df = loan
annual_inc_df.categorised_annual_inc = annual_inc_df.categorised_annual_inc.astype('cate
annual_inc_df.categorised_annual_inc.cat.set_categories(order_category, inplace = True)
annual_inc_df.sort_values(['categorised_annual_inc'])

annual_inc_crosstab = pd.crosstab(annual_inc_df['categorised_annual_inc'],annual_inc_df[
annual_inc_crosstab.drop(annual_inc_crosstab.tail(1).index, inplace = True)
annual_inc_crosstab['percentage_defaulted'] = round(100*((annual_inc_crosstab['Charged O
display(annual_inc_crosstab)
plot_map(annual_inc_crosstab, 'Annual income', 'annual income', .10)
```

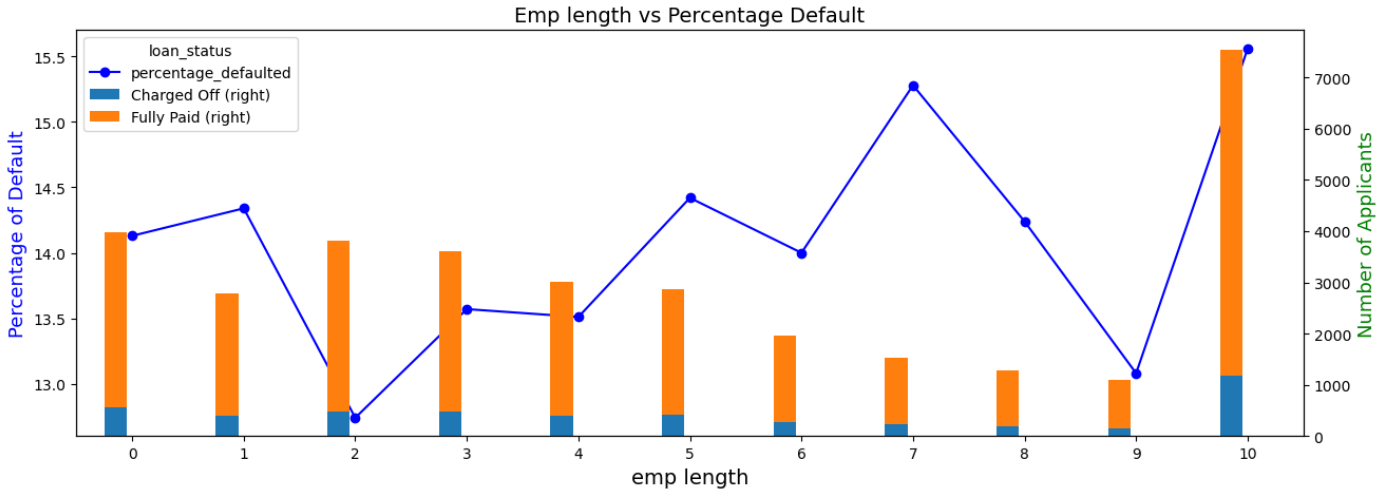
loan_status	Charged Off	Fully Paid	All	percentage_defaulted
categorised_annual_inc				
low income	1406	6686	8092	17.38
medium income	1247	7239	8486	14.69
high income	1208	7414	8622	14.01
very high income	907	7385	8292	10.94



In [119…

```
emp_length_df = loan
emp_length_df.sort_values(['emp_length'])


emp_length_crosstab = pd.crosstab(emp_length_df['emp_length'],emp_length_df['loan_status
emp_length_crosstab.drop(emp_length_crosstab.tail(1).index, inplace = True)
emp_length_crosstab['percentage_defaulted'] = round(100*((emp_length_crosstab['Charged O
display(emp_length_crosstab)

plot_map(emp_length_crosstab, 'Emp length', 'emp length', .20)
```

loan_status	Charged Off	Fully Paid	All	percentage_defaulted
emp_length				
0	561	3408	3969	14.13
1	400	2389	2789	14.34
2	485	3321	3806	12.74
3	489	3115	3604	13.57
4	406	2600	3006	13.51
5	414	2458	2872	14.42
6	276	1696	1972	14.00
7	235	1303	1538	15.28
8	184	1108	1292	14.24
9	144	957	1101	13.08
10	1174	6369	7543	15.56



Following are main parameters, taken into consideration for arriving at the analysis conclusion.

*Interest rate*Purpose *Grade*Term *Emp Length*Annual Income

As per the analysis it can inferred that applicants who are of low income and have taken high interest loan with longer duration for small business have more profitability of defaulting.

Hence Extra care should be taken before lending them loan.

In [ ]:

In [ ]: