Banking Loan

Importing Libraries and warnings

In [2]:
```
import warnings
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing data

In [86]:
```
warnings.filterwarnings("ignore")
loan = pd.read_csv(r"C:\Users\SURBDESA\Downloads\loan (1)\loan.csv")
```

In [87]:
```
loan
```

Out[87]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	num_tl_90g_dpd_24m	nun
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	B2	...	NaN	
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	C4	...	NaN	
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	C5	...	NaN	
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	C1	...	NaN	
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	B5	...	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	
39712	92187	92174	2500	2500	1075.0	36 months	8.07%	78.42	A	A4	...	NaN	
39713	90665	90607	8500	8500	875.0	36 months	10.28%	275.38	C	C1	...	NaN	
39714	90395	90390	5000	5000	1325.0	36 months	8.07%	156.84	A	A4	...	NaN	
39715	90376	89243	5000	5000	650.0	36 months	7.43%	155.38	A	A2	...	NaN	
39716	87023	86999	7500	7500	800.0	36 months	13.75%	255.43	E	E2	...	NaN	

39717 rows × 111 columns

Dropping columns where data missing is greater than 90%

In [88]:

```
missing_columns = loan.columns[100*(loan.isnull().sum()/len(loan.index)) > 90]
print(missing_columns)
```

```
Index(['mths_since_last_record', 'next_pymnt_d', 'mths_since_last_major_derog',
       'annual_inc_joint', 'dti_joint', 'verification_status_joint',
       'tot_coll_amt', 'tot_cur_bal', 'open_acc_6m', 'open_il_6m',
       'open_il_12m', 'open_il_24m', 'mths_since_rcnt_il', 'total_bal_il',
       'il_util', 'open_rv_12m', 'open_rv_24m', 'max_bal_bc', 'all_util',
       'total_rev_hi_lim', 'inq_fi', 'total_cu_tl', 'inq_last_12m',
       'acc_open_past_24mths', 'avg_cur_bal', 'bc_open_to_buy', 'bc_util',
       'mo_sin_old_il_acct', 'mo_sin_old_rev_tl_op', 'mo_sin_rcnt_rev_tl_op',
       'mo_sin_rcnt_tl', 'mort_acc', 'mths_since_recent_bc',
       'mths_since_recent_bc_dlq', 'mths_since_recent_inq',
       'mths_since_recent_revol_delinq', 'num_accts_ever_120_pd',
       'num_actv_bc_tl', 'num_actv_rev_tl', 'num_bc_sats', 'num_bc_tl',
       'num_il_tl', 'num_op_rev_tl', 'num_rev_accts', 'num_rev_tl_bal_gt_0',
       'num_sats', 'num_tl_120dpd_2m', 'num_tl_30dpd', 'num_tl_90g_dpd_24m',
       'num_tl_op_past_12m', 'pct_tl_nvr_dlq', 'percent_bc_gt_75',
       'tot_hi_cred_lim', 'total_bal_ex_mort', 'total_bc_limit',
       'total_il_high_credit_limit'],
      dtype='object')
```

In [89]:
```
loan.drop(missing_columns , axis = 1, inplace = True)
```

In [90]:
```
loan.isnull().sum()
```

Out[90]:

```
id                            0
member_id                     0
loan_amnt                     0
funded_amnt                   0
funded_amnt_inv               0
term                          0
int_rate                      0
installment                   0
grade                         0
sub_grade                     0
emp_title                  2459
emp_length                 1075
home_ownership                0
annual_inc                    0
verification_status           0
issue_d                       0
loan_status                   0
pymnt_plan                    0
url                           0
desc                      12940
purpose                       0
title                        11
zip_code                      0
addr_state                    0
dti                           0
delinq_2yrs                   0
earliest_cr_line              0
inq_last_6mths                0
mths_since_last_delinq    25682
open_acc                      0
pub_rec                       0
revol_bal                     0
revol_util                   50
total_acc                     0
initial_list_status           0
out_prncp                     0
out_prncp_inv                 0
total_pymnt                   0
total_pymnt_inv               0
total_rec_prncp               0
total_rec_int                 0
total_rec_late_fee            0
recoveries                    0
collection_recovery_fee       0
```

```
last_pymnt_d                  71
last_pymnt_amnt                0
last_credit_pull_d             2
collections_12_mths_ex_med    56
policy_code                    0
application_type               0
acc_now_delinq                 0
chargeoff_within_12_mths      56
delinq_amnt                    0
pub_rec_bankruptcies         697
tax_liens                     39
dtype: int64
```

In [91]:

```
# dropping the two columns
loan = loan.drop(['desc', 'mths_since_last_delinq'], axis=1)
```

In [92]:

```
loan.isnull().sum()
```

Out[92]:
```
id                            0
member_id                     0
loan_amnt                     0
funded_amnt                   0
funded_amnt_inv               0
term                          0
int_rate                      0
installment                   0
grade                         0
sub_grade                     0
emp_title                  2459
emp_length                 1075
home_ownership                0
annual_inc                    0
verification_status           0
issue_d                       0
loan_status                   0
pymnt_plan                    0
url                           0
purpose                       0
title                        11
zip_code                      0
addr_state                    0
dti                           0
delinq_2yrs                   0
earliest_cr_line              0
inq_last_6mths                0
open_acc                      0
pub_rec                       0
revol_bal                     0
revol_util                   50
total_acc                     0
initial_list_status           0
out_prncp                     0
out_prncp_inv                 0
total_pymnt                   0
total_pymnt_inv               0
total_rec_prncp               0
total_rec_int                 0
total_rec_late_fee            0
recoveries                    0
collection_recovery_fee       0
last_pymnt_d                 71
last_pymnt_amnt               0
```

```
last_credit_pull_d              2
collections_12_mths_ex_med     56
policy_code                     0
application_type                0
acc_now_delinq                  0
chargeoff_within_12_mths       56
delinq_amnt                     0
pub_rec_bankruptcies          697
tax_liens                      39
dtype: int64
```

In [93]:

```
loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39717 entries, 0 to 39716
Data columns (total 53 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   id                          39717 non-null  int64  
 1   member_id                   39717 non-null  int64  
 2   loan_amnt                   39717 non-null  int64  
 3   funded_amnt                 39717 non-null  int64  
 4   funded_amnt_inv             39717 non-null  float64
 5   term                        39717 non-null  object 
 6   int_rate                    39717 non-null  object 
 7   installment                 39717 non-null  float64
 8   grade                       39717 non-null  object 
 9   sub_grade                   39717 non-null  object 
 10  emp_title                   37258 non-null  object 
 11  emp_length                  38642 non-null  object 
 12  home_ownership              39717 non-null  object 
 13  annual_inc                  39717 non-null  float64
 14  verification_status         39717 non-null  object 
 15  issue_d                     39717 non-null  object 
 16  loan_status                 39717 non-null  object 
 17  pymnt_plan                  39717 non-null  object 
 18  url                         39717 non-null  object 
 19  purpose                     39717 non-null  object 
 20  title                       39706 non-null  object 
 21  zip_code                    39717 non-null  object 
 22  addr_state                  39717 non-null  object 
 23  dti                         39717 non-null  float64
 24  delinq_2yrs                 39717 non-null  int64  
 25  earliest_cr_line            39717 non-null  object 
 26  inq_last_6mths              39717 non-null  int64  
 27  open_acc                    39717 non-null  int64  
 28  pub_rec                     39717 non-null  int64  
 29  revol_bal                   39717 non-null  int64  
 30  revol_util                  39667 non-null  object 
 31  total_acc                   39717 non-null  int64  
 32  initial_list_status         39717 non-null  object 
 33  out_prncp                   39717 non-null  float64
 34  out_prncp_inv               39717 non-null  float64
 35  total_pymnt                 39717 non-null  float64
 36  total_pymnt_inv             39717 non-null  float64
 37  total_rec_prncp             39717 non-null  float64
 38  total_rec_int               39717 non-null  float64
```

```
 39  total_rec_late_fee          39717 non-null  float64
 40  recoveries                  39717 non-null  float64
 41  collection_recovery_fee     39717 non-null  float64
 42  last_pymnt_d                39646 non-null  object
 43  last_pymnt_amnt             39717 non-null  float64
 44  last_credit_pull_d          39715 non-null  object
 45  collections_12_mths_ex_med  39661 non-null  float64
 46  policy_code                 39717 non-null  int64
 47  application_type            39717 non-null  object
 48  acc_now_delinq              39717 non-null  int64
 49  chargeoff_within_12_mths    39661 non-null  float64
 50  delinq_amnt                 39717 non-null  int64
 51  pub_rec_bankruptcies        39020 non-null  float64
 52  tax_liens                   39678 non-null  float64
dtypes: float64(18), int64(13), object(22)
memory usage: 16.1+ MB
```

We can clearly see that employee length that is how many years employee has worked has object data type

We can convert data to integer variable for better data quality

In [94]:
```
loan['emp_length'].value_counts()
```

Out[94]:
```
10+ years    8879
< 1 year     4583
2 years      4388
3 years      4095
4 years      3436
5 years      3282
1 year       3240
6 years      2229
7 years      1773
8 years      1479
9 years      1258
Name: emp_length, dtype: int64
```

In [95]:
```
loan['months'] = loan.term.apply(lambda x : x.split()[0])
```

In [96]:
```
loan.drop(['term'],axis = 1,inplace = True)
```

In [97]:
```
loan = loan[~loan.emp_length.isnull()]
```

In [98]:
```
loan.emp_length.isnull().sum()
```

Out[98]:
```
0
```

In [99]:
```
loan.emp_length.value_counts()
```

Out[99]:
```
10+ years    8879
< 1 year     4583
2 years      4388
3 years      4095
4 years      3436
5 years      3282
1 year       3240
6 years      2229
7 years      1773
8 years      1479
9 years      1258
Name: emp_length, dtype: int64
```

In [100…
```
loan['emp_length'] = loan.emp_length.str.extract('(\d+)')
```

In [101…
```
loan.emp_length.value_counts()
```

Out[101]:
```
10    8879
1     7823
2     4388
3     4095
4     3436
5     3282
6     2229
7     1773
8     1479
9     1258
Name: emp_length, dtype: int64
```

In [102…
```
loan['emp_length'] = loan['emp_length'].apply(lambda x: pd.to_numeric(x))
```

In [103…
```
loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 38642 entries, 0 to 39716
Data columns (total 53 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   id                          38642 non-null  int64  
 1   member_id                   38642 non-null  int64  
 2   loan_amnt                   38642 non-null  int64  
 3   funded_amnt                 38642 non-null  int64  
 4   funded_amnt_inv             38642 non-null  float64
 5   int_rate                    38642 non-null  object 
 6   installment                 38642 non-null  float64
 7   grade                       38642 non-null  object 
 8   sub_grade                   38642 non-null  object 
 9   emp_title                   37202 non-null  object 
 10  emp_length                  38642 non-null  int64  
 11  home_ownership              38642 non-null  object 
 12  annual_inc                  38642 non-null  float64
 13  verification_status         38642 non-null  object 
 14  issue_d                     38642 non-null  object 
 15  loan_status                 38642 non-null  object 
 16  pymnt_plan                  38642 non-null  object 
 17  url                         38642 non-null  object 
 18  purpose                     38642 non-null  object 
 19  title                       38632 non-null  object 
 20  zip_code                    38642 non-null  object 
 21  addr_state                  38642 non-null  object 
 22  dti                         38642 non-null  float64
 23  delinq_2yrs                 38642 non-null  int64  
 24  earliest_cr_line            38642 non-null  object 
 25  inq_last_6mths              38642 non-null  int64  
 26  open_acc                    38642 non-null  int64  
 27  pub_rec                     38642 non-null  int64  
 28  revol_bal                   38642 non-null  int64  
 29  revol_util                  38595 non-null  object 
 30  total_acc                   38642 non-null  int64  
 31  initial_list_status         38642 non-null  object 
 32  out_prncp                   38642 non-null  float64
 33  out_prncp_inv               38642 non-null  float64
 34  total_pymnt                 38642 non-null  float64
 35  total_pymnt_inv             38642 non-null  float64
 36  total_rec_prncp             38642 non-null  float64
 37  total_rec_int               38642 non-null  float64
 38  total_rec_late_fee          38642 non-null  float64
```

```
 39  recoveries                  38642 non-null  float64
 40  collection_recovery_fee     38642 non-null  float64
 41  last_pymnt_d                38576 non-null  object
 42  last_pymnt_amnt             38642 non-null  float64
 43  last_credit_pull_d          38640 non-null  object
 44  collections_12_mths_ex_med  38586 non-null  float64
 45  policy_code                 38642 non-null  int64
 46  application_type            38642 non-null  object
 47  acc_now_delinq              38642 non-null  int64
 48  chargeoff_within_12_mths    38586 non-null  float64
 49  delinq_amnt                 38642 non-null  int64
 50  pub_rec_bankruptcies        37945 non-null  float64
 51  tax_liens                   38603 non-null  float64
 52  months                      38642 non-null  object
dtypes: float64(18), int64(14), object(21)
memory usage: 15.9+ MB
```

Data analysis

The objective is to identify predictors of default so that at the time of loan application, we can use those variables for approval/rejection of the loan

In [104…

```
behaviour_var =  [
  "delinq_2yrs",
  "earliest_cr_line",
  "inq_last_6mths",
  "open_acc",
  "pub_rec",
  "revol_bal",
  "revol_util",
  "total_acc",
  "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",
  "last_credit_pull_d",
  "application_type"]
behaviour_var
```

Out[104]:
```
['delinq_2yrs',
 'earliest_cr_line',
 'inq_last_6mths',
 'open_acc',
 'pub_rec',
 'revol_bal',
 'revol_util',
 'total_acc',
 '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',
 'last_credit_pull_d',
 'application_type']
```

In [105…
```
loan.drop(behaviour_var, axis = 1 , inplace = True)
```

In [106…
```
loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 38642 entries, 0 to 39716
Data columns (total 32 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   id                          38642 non-null  int64  
 1   member_id                   38642 non-null  int64  
 2   loan_amnt                   38642 non-null  int64  
 3   funded_amnt                 38642 non-null  int64  
 4   funded_amnt_inv             38642 non-null  float64
 5   int_rate                    38642 non-null  object 
 6   installment                 38642 non-null  float64
 7   grade                       38642 non-null  object 
 8   sub_grade                   38642 non-null  object 
 9   emp_title                   37202 non-null  object 
 10  emp_length                  38642 non-null  int64  
 11  home_ownership              38642 non-null  object 
 12  annual_inc                  38642 non-null  float64
 13  verification_status         38642 non-null  object 
 14  issue_d                     38642 non-null  object 
 15  loan_status                 38642 non-null  object 
 16  pymnt_plan                  38642 non-null  object 
 17  url                         38642 non-null  object 
 18  purpose                     38642 non-null  object 
 19  title                       38632 non-null  object 
 20  zip_code                    38642 non-null  object 
 21  addr_state                  38642 non-null  object 
 22  dti                         38642 non-null  float64
 23  initial_list_status         38642 non-null  object 
 24  collections_12_mths_ex_med  38586 non-null  float64
 25  policy_code                 38642 non-null  int64  
 26  acc_now_delinq              38642 non-null  int64  
 27  chargeoff_within_12_mths    38586 non-null  float64
 28  delinq_amnt                 38642 non-null  int64  
 29  pub_rec_bankruptcies        37945 non-null  float64
 30  tax_liens                   38603 non-null  float64
 31  months                      38642 non-null  object 
dtypes: float64(8), int64(8), object(16)
memory usage: 9.7+ MB
```

In [107…

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

In [108…

```
loan.loan_status.value_counts()
```

Out[108]:
```
Fully Paid     32145
Charged Off     5399
Current         1098
Name: loan_status, dtype: int64
```

Loan status has 3 categories , there is no need of current category in data as we need to find defaulters based on historical loan application status.

In [109…
```
loan = loan[loan['loan_status'] != 'Current']
loan['loan_status'] = loan['loan_status'].apply(lambda x: 'Non-Default' if x=='Fully Paid' else 'Default')
```

In [110…
```
loan['loan_status_categor'] = loan.loan_status
```

In [111…
```
loan.loan_status_categor.value_counts()
```

Out[111]:
```
Non-Default    32145
Default         5399
Name: loan_status_categor, dtype: int64
```

In [112…
```
#Current loan status is not needed for analyzing
loan = loan[loan['loan_status'] != 'Current']
loan['loan_status'] = loan['loan_status'].apply(lambda x: 0 if x=='Non-Default' else 1)

# converting loan_status to integer type
loan['loan_status'] = loan['loan_status'].apply(lambda x: pd.to_numeric(x))

# summarising the values
loan['loan_status'].value_counts()
```

Out[112]:
```
0    32145
1     5399
Name: loan_status, dtype: int64
```

Default rate

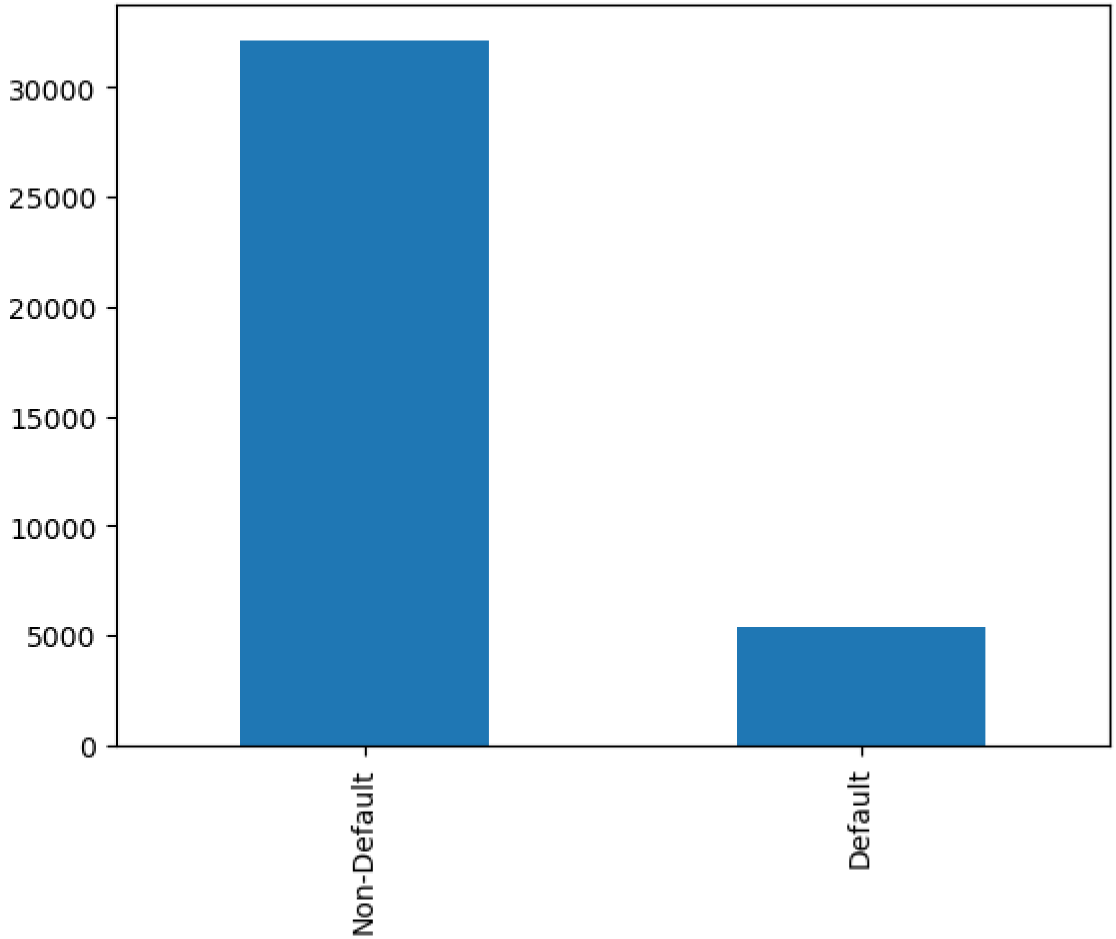
In [113…
```
round(np.mean(loan['loan_status']==1), 2)
```

Out[113]:
```
0.14
```

Loan default rate is 14% .

In [114…
```
loan.loan_status_categor.value_counts().plot.bar()
```

Out[114]: <AxesSubplot:>



In [115…
```
loan
```

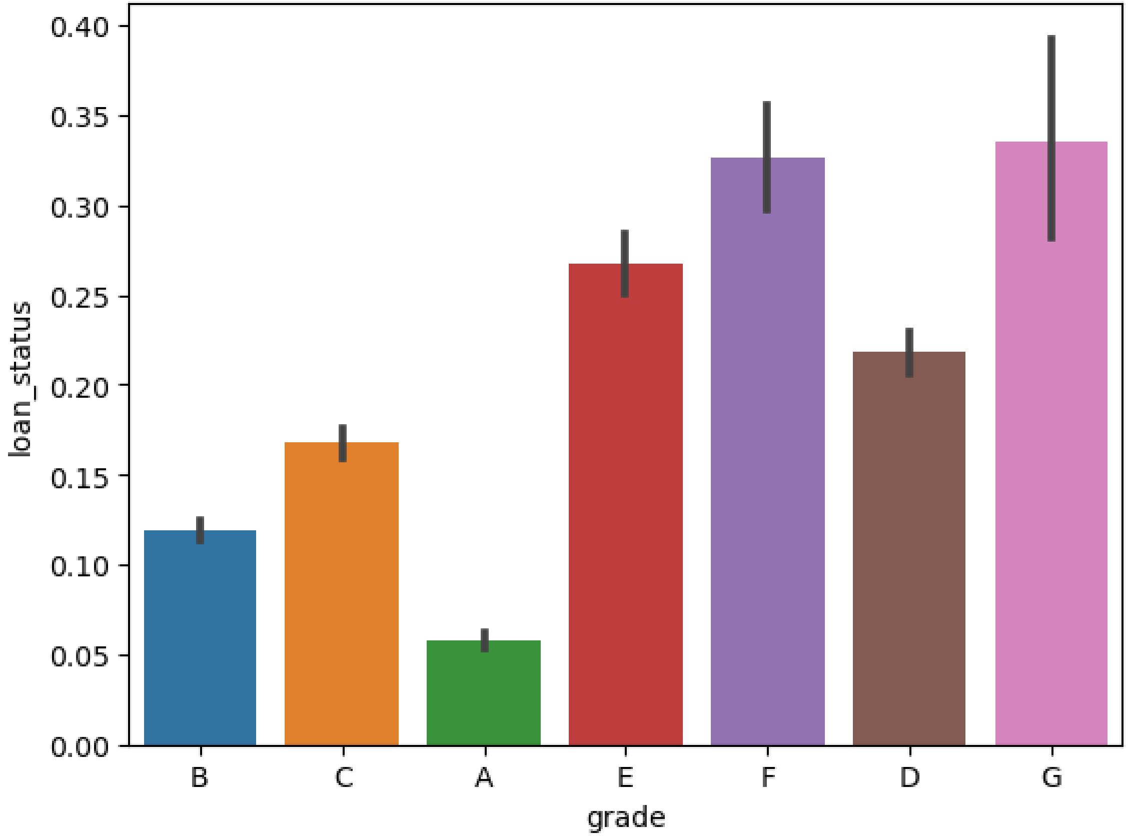
Out[115]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	installment	grade	sub_grade	emp_title	...	initial_list_status	col
0	1077501	1296599	5000	5000	4975.0	10.65%	162.87	B	B2	NaN	...	f	
1	1077430	1314167	2500	2500	2500.0	15.27%	59.83	C	C4	Ryder	...	f	
2	1077175	1313524	2400	2400	2400.0	15.96%	84.33	C	C5	NaN	...	f	
3	1076863	1277178	10000	10000	10000.0	13.49%	339.31	C	C1	AIR RESOURCES BOARD	...	f	
5	1075269	1311441	5000	5000	5000.0	7.90%	156.46	A	A4	Veolia Transportaton	...	f	
...	...	...	...	...	...	...	...	...	...	...	...	...	
39712	92187	92174	2500	2500	1075.0	8.07%	78.42	A	A4	FiSite Research	...	f	
39713	90665	90607	8500	8500	875.0	10.28%	275.38	C	C1	Squarewave Solutions, Ltd.	...	f	
39714	90395	90390	5000	5000	1325.0	8.07%	156.84	A	A4	NaN	...	f	
39715	90376	89243	5000	5000	650.0	7.43%	155.38	A	A2	NaN	...	f	
39716	87023	86999	7500	7500	800.0	13.75%	255.43	E	E2	Evergreen Center	...	f	

37544 rows × 29 columns

In [116…

```
sns.barplot(x='grade', y='loan_status', data=loan)
plt.show()
```

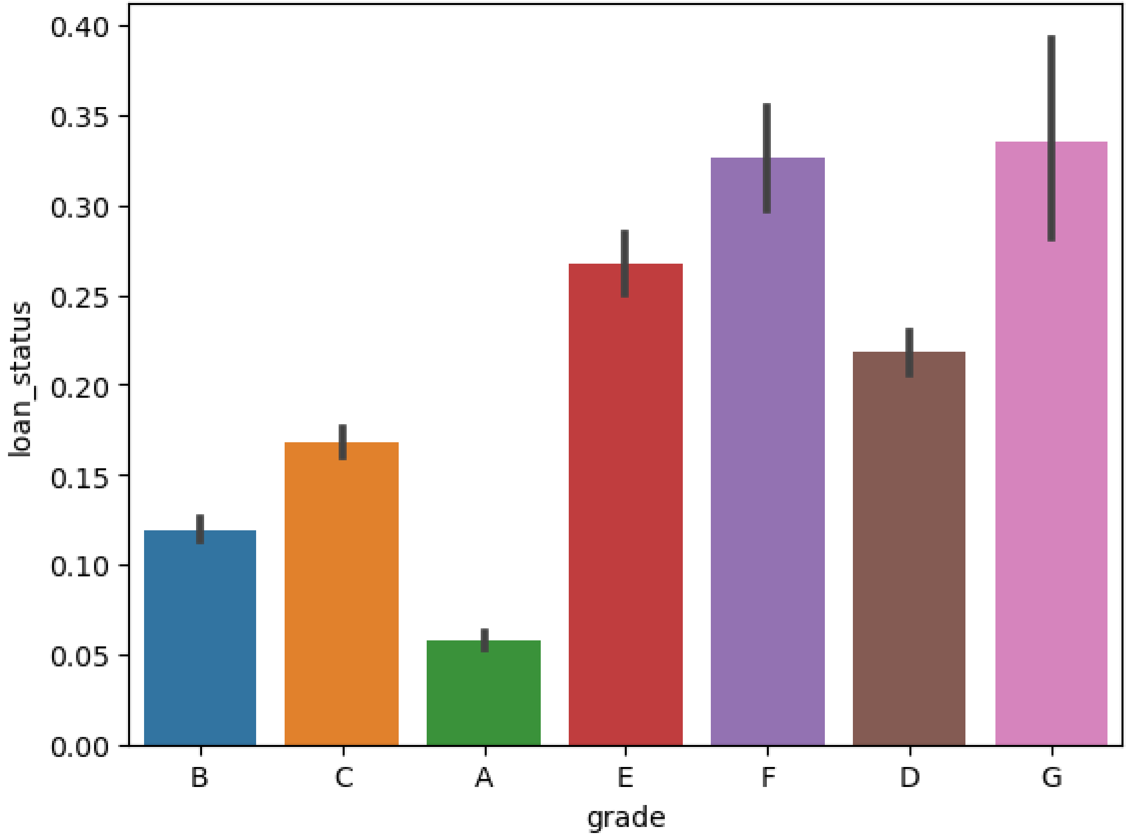


Overall default rate is 14%

So let's plot against categorical variable to gain more insight

In [117…

```
sns.barplot(x='grade', y='loan_status', data=loan)
plt.show()
```



In [118…

```
loan.grade.value_counts()
```

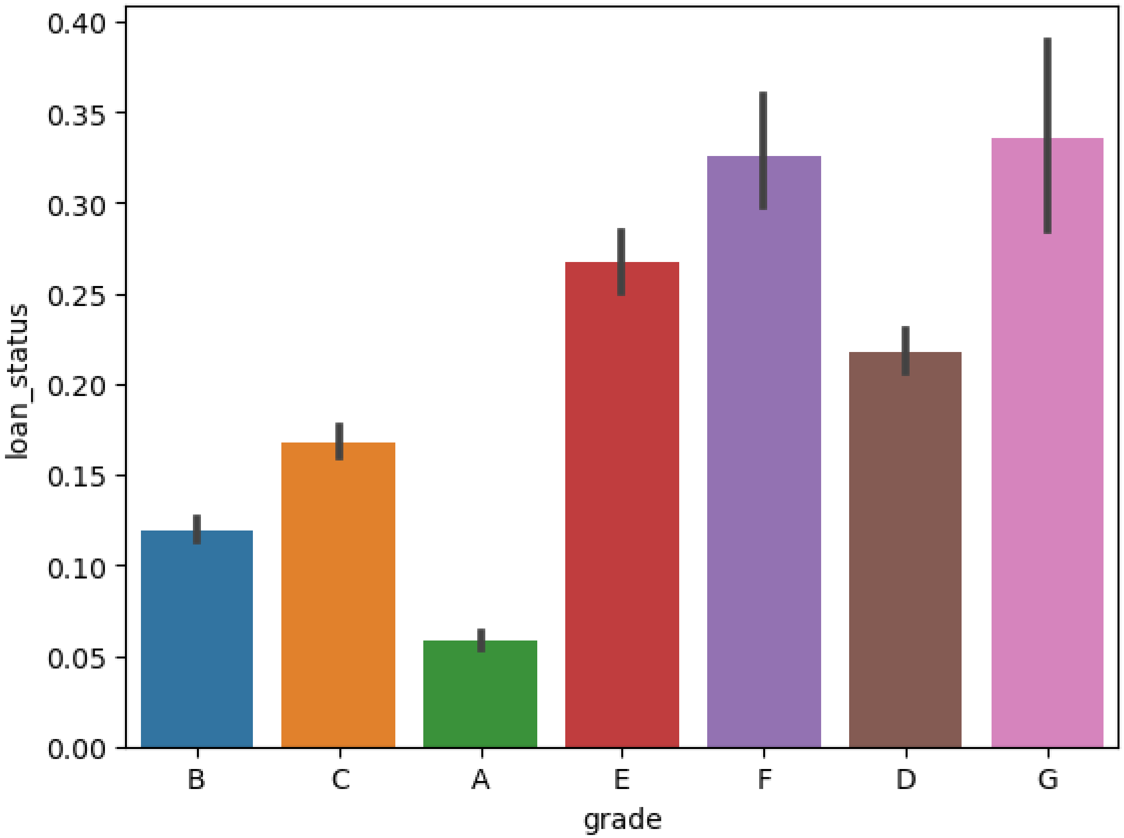
Out[118]:

```
B    11359
A     9660
C     7669
D     4979
E     2620
F      959
G      298
Name: grade, dtype: int64
```

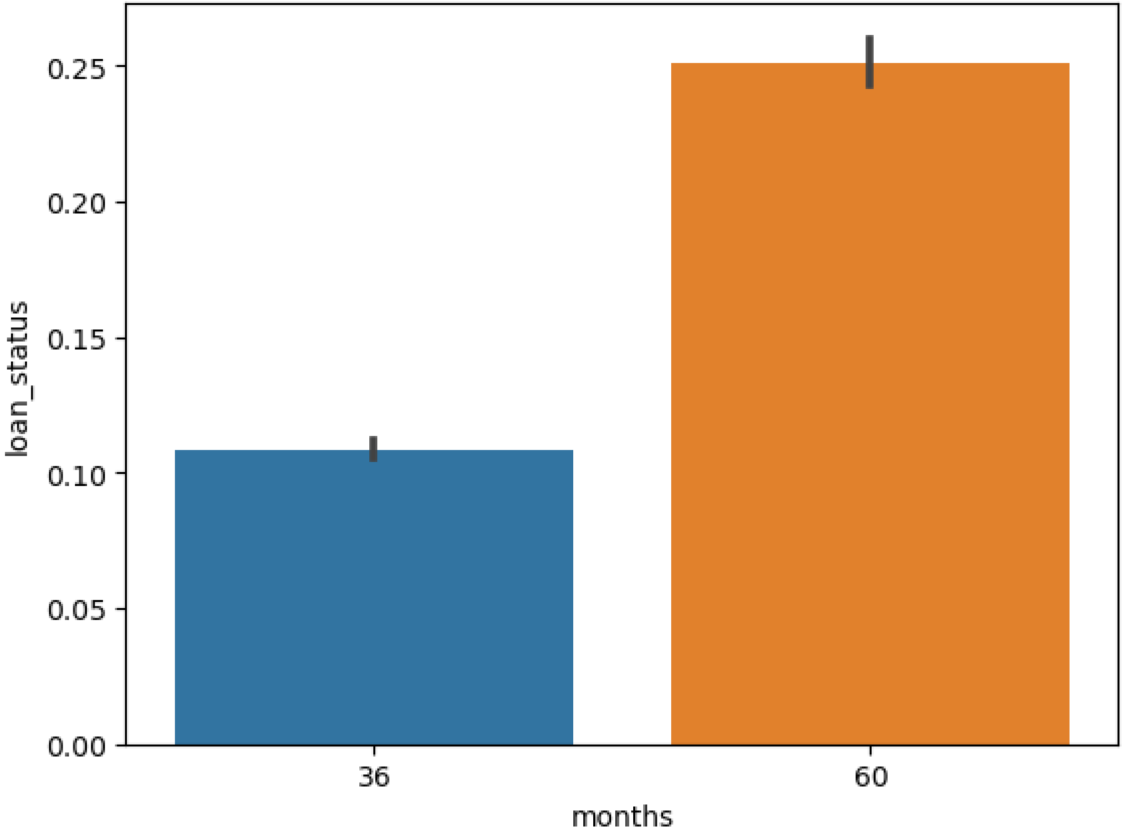
In [119…

```
def plot_data(categor_var):
    sns.barplot(x=categor_var, y='loan_status', data=loan)
    plt.show()
```

In [120…
```
# We can clearly figure out that grade E,F,D,G has higher default rate.
plot_data('grade')
```

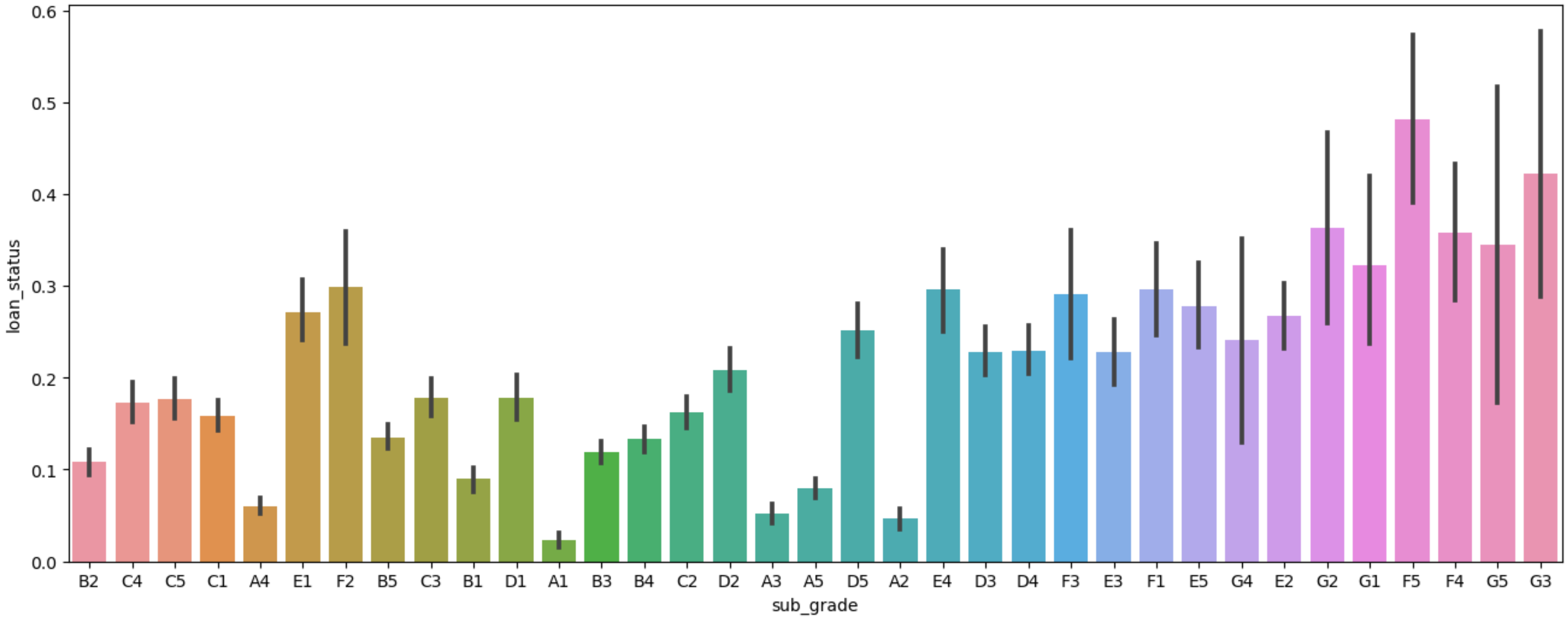


In [121…
```
# we can clearly see that 60 months term has higher default rate than 36 months.
plot_data('months')
```



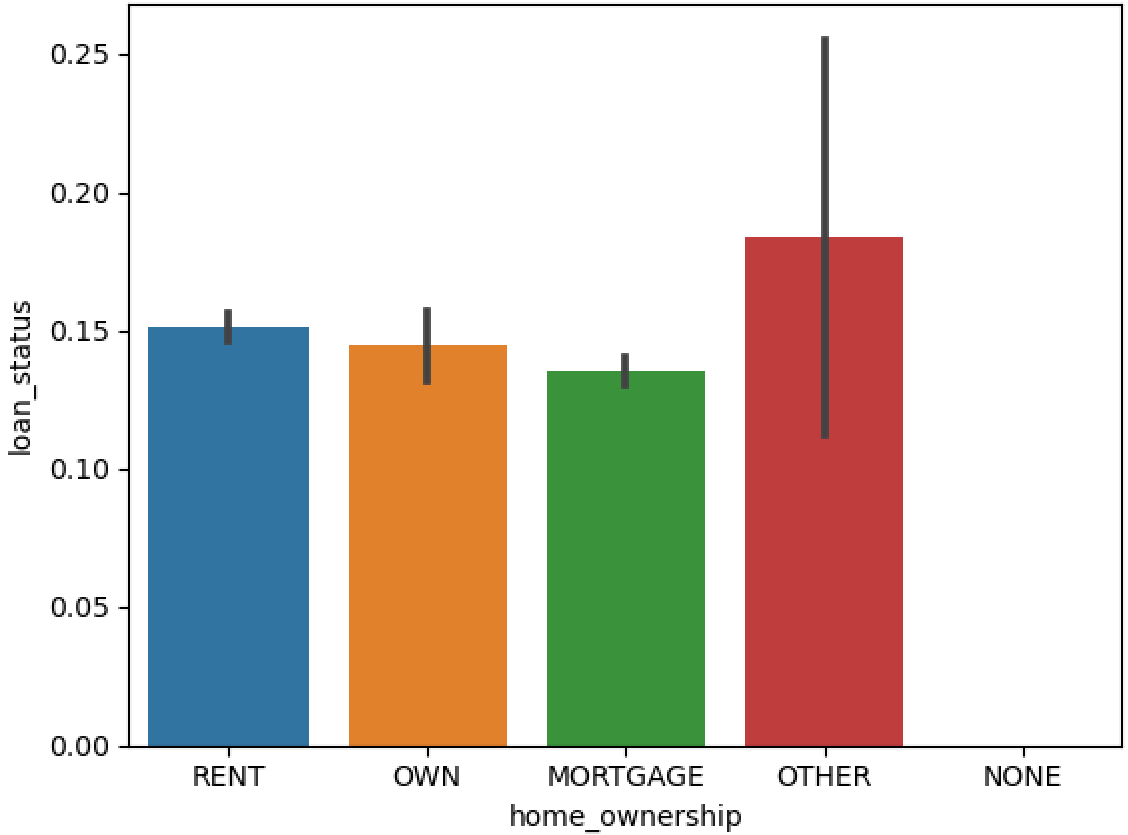
In [122…

```
# sub-grade:  A1 is better than A2 better than A3 and so on
plt.figure(figsize=(16, 6))
plot_data('sub_grade')
```



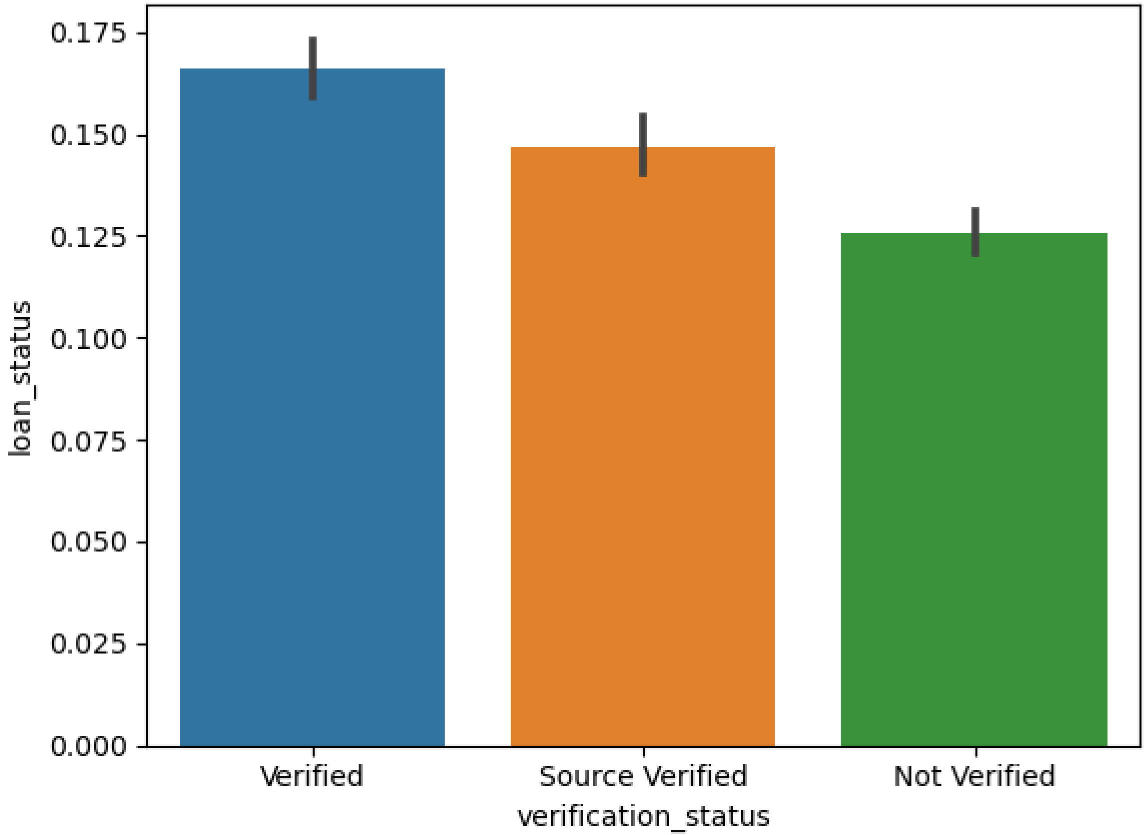
In [123…

```
plot_data('home_ownership')
```



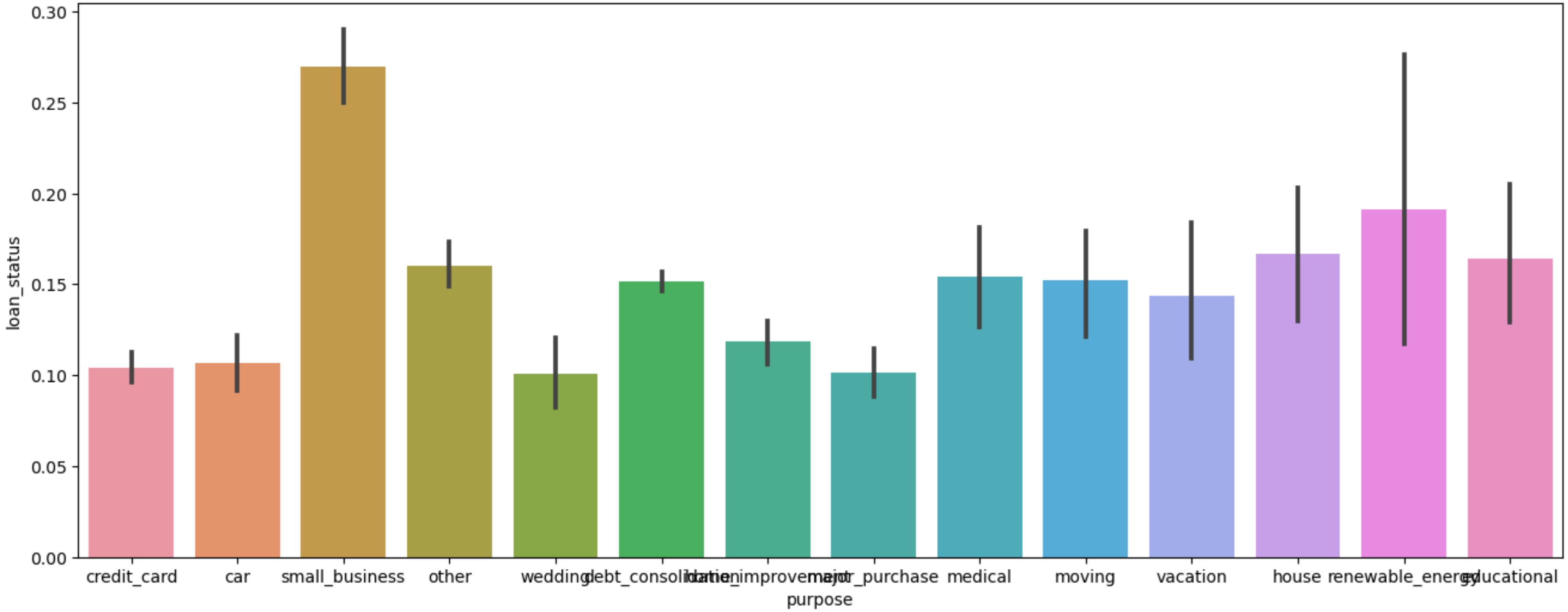
In [125…

```
# verification_status: Verified loan has more default rate than not verified
plot_data('verification_status')
```



In [126…

```
# purpose: small business loans defualt rate is the largest than compared to others.
plt.figure(figsize=(16, 6))
plot_data('purpose')
```



In [127…
```
loan['issue_d'].describe
```

Out[127]:
```
<bound method NDFrame.describe of 0        Dec-11
1         Dec-11
2         Dec-11
3         Dec-11
5         Dec-11
           ...
39712     Jul-07
39713     Jul-07
39714     Jul-07
39715     Jul-07
39716     Jun-07
Name: issue_d, Length: 37544, dtype: object>
```

In [128…
```
loan['Dates'] = pd.to_datetime(loan['issue_d'], format='%b-%y')
```

In [129…
```
loan['month'] = loan['Dates'].apply(lambda x: x.month)
```

In [130…
```
loan['month'].describe
```

Out[130]:
```
<bound method NDFrame.describe of 0        12
1        12
2        12
3        12
5        12
         ..
39712     7
39713     7
39714     7
39715     7
39716     6
Name: month, Length: 37544, dtype: int64>
```

In [131…
```
loan['year'] = loan['Dates'].apply(lambda x: x.year)
```

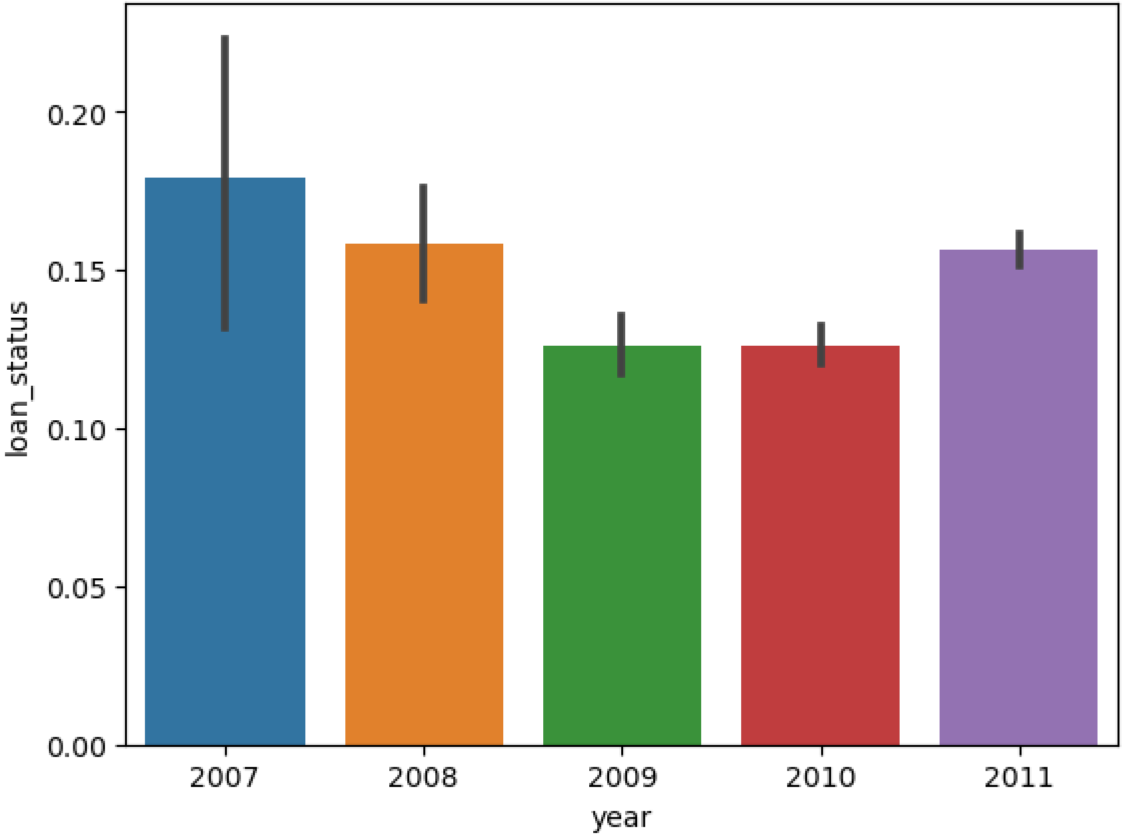
In [133…
```
loan['year'].value_counts()
```

Out[133]:
```
2011    19801
2010    11214
2009     4716
2008     1562
2007      251
Name: year, dtype: int64
```

In [134…
```
loan.groupby('year').year.count()
```

Out[134]:
```
year
2007      251
2008     1562
2009     4716
2010    11214
2011    19801
Name: year, dtype: int64
```

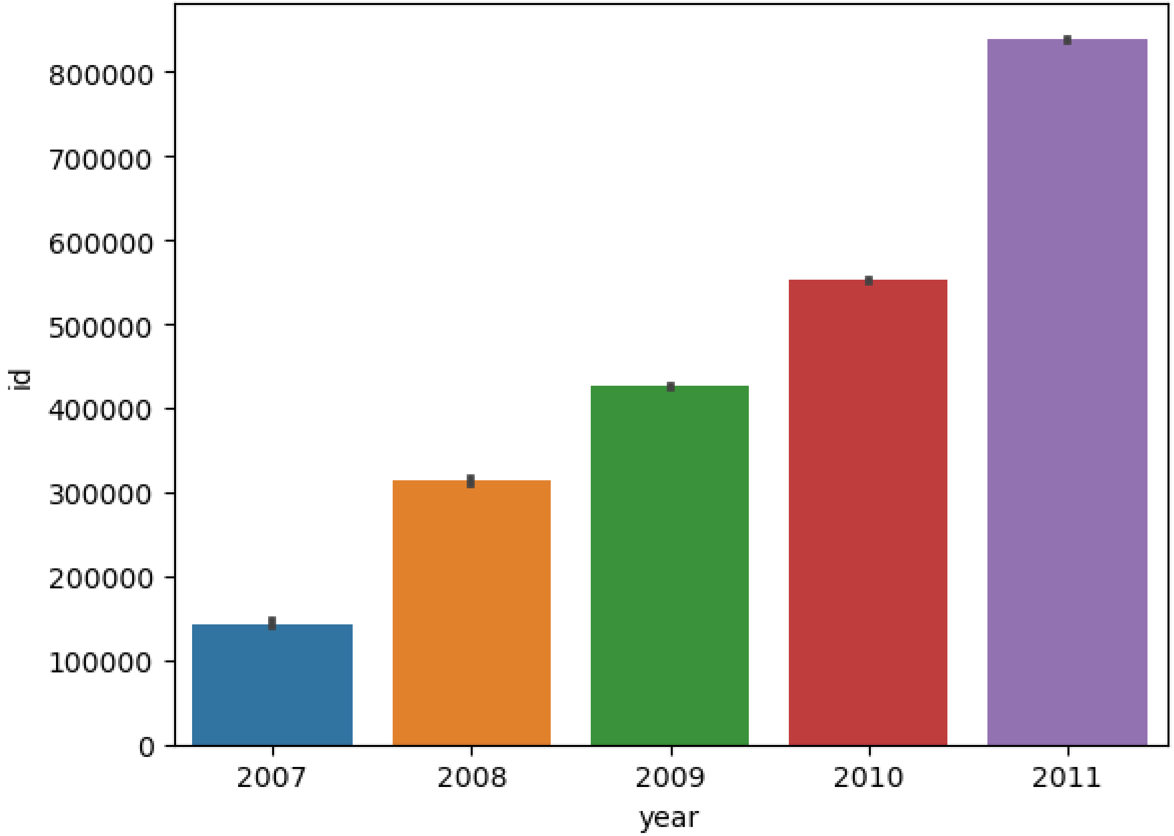
In [135…
```
#the default rate was high in the year 2007 then gradually decreased but suddenly got increased in the year 2011
plot_data('year')
```



In [137…

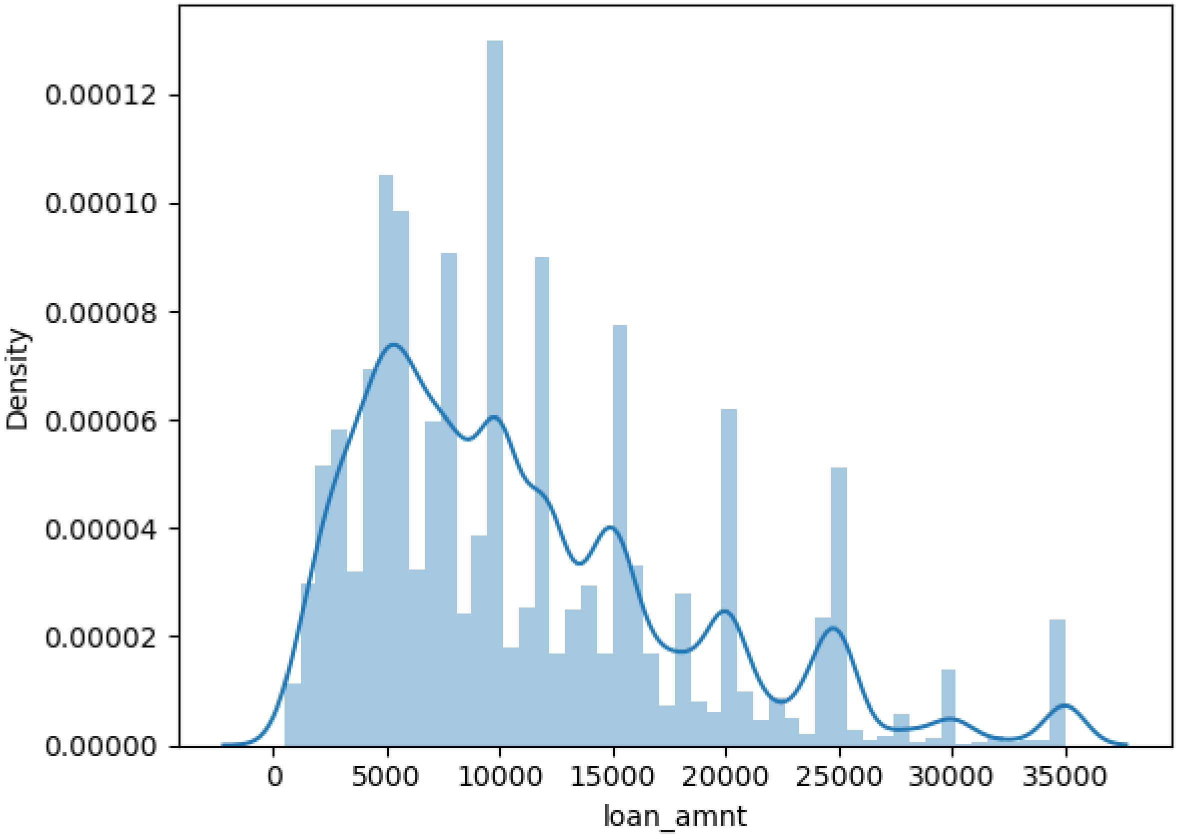
```
sns.barplot(x='year', y='id', data=loan)
```

Out[137]: <AxesSubplot:xlabel='year', ylabel='id'>



In [204…

```
sns.distplot(loan['loan_amnt'])
plt.show()
```



In [213…

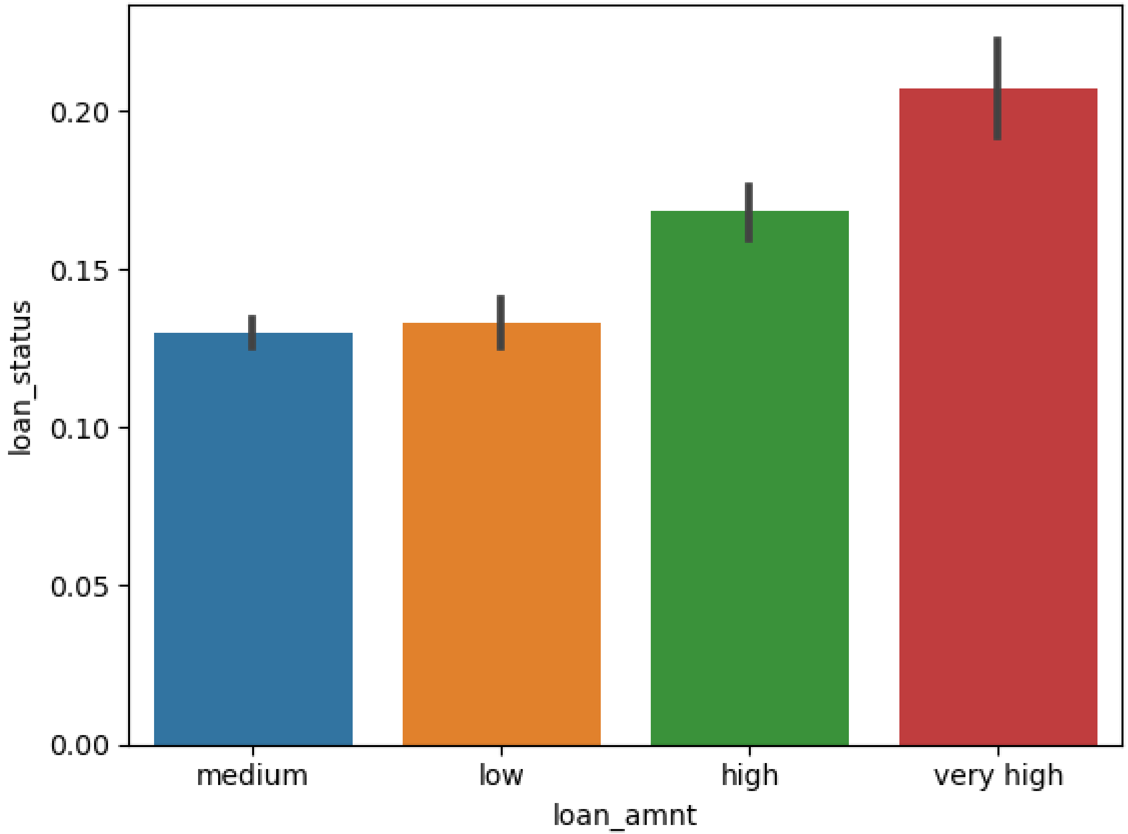
```
#Let's bin the loan amount variable into small, medium, high, very high.
def loan_amount(n):
    if n < 5000:
        return 'low'
    elif n >=5000 and n < 15000:
        return 'medium'
    elif n >= 15000 and n < 25000:
        return 'high'
    else:
        return 'very high'

loan['loan_amnt'] = loan['loan_amnt'].apply(lambda x: loan_amount(x))
```

In [214…

```
loan['loan_amnt'].value_counts()
```

Out[214]:
```
medium       20157
high          7572
low           7095
very high     2720
Name: loan_amnt, dtype: int64
```

In [216…
```
# higher the loan amount, higher is the default rate
plot_data('loan_amnt')
```



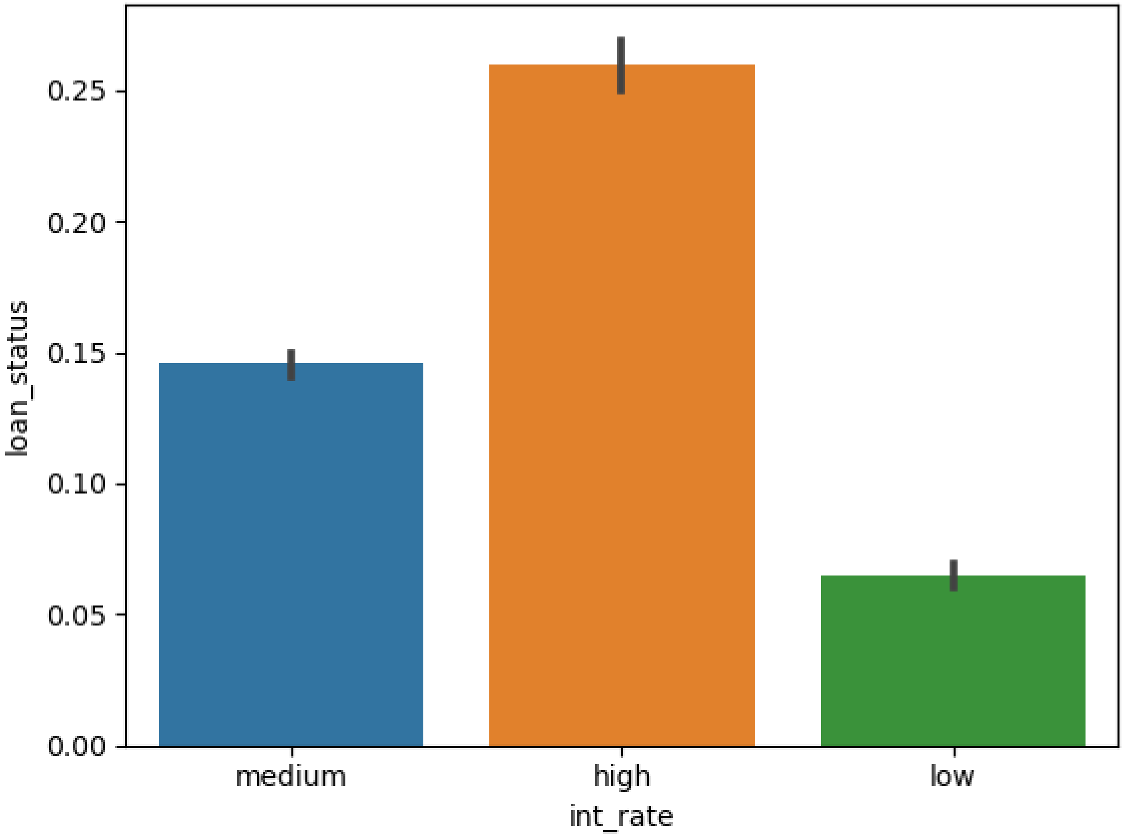
In [227…
```
loan['int_rate'] = loan['int_rate'].apply(lambda x: pd.to_numeric(x.split("%")[0]))
```

In [228…
```
loan.int_rate.value_counts()
```

Out[228]:
```
10.99    891
11.49    766
7.51     756
13.49    736
7.88     701
        ...
16.96      1
18.36      1
16.15      1
16.01      1
16.20      1
Name: int_rate, Length: 370, dtype: int64
```

In [229…
```
def int_rate(n):
    if n <= 10:
        return 'low'
    elif n > 10 and n <=15:
        return 'medium'
    else:
        return 'high'


loan['int_rate'] = loan['int_rate'].apply(lambda x: int_rate(x))
```

In [230…
```
#Higher the rate of interest higher is the default rate
plot_data('int_rate')
```



In [233…

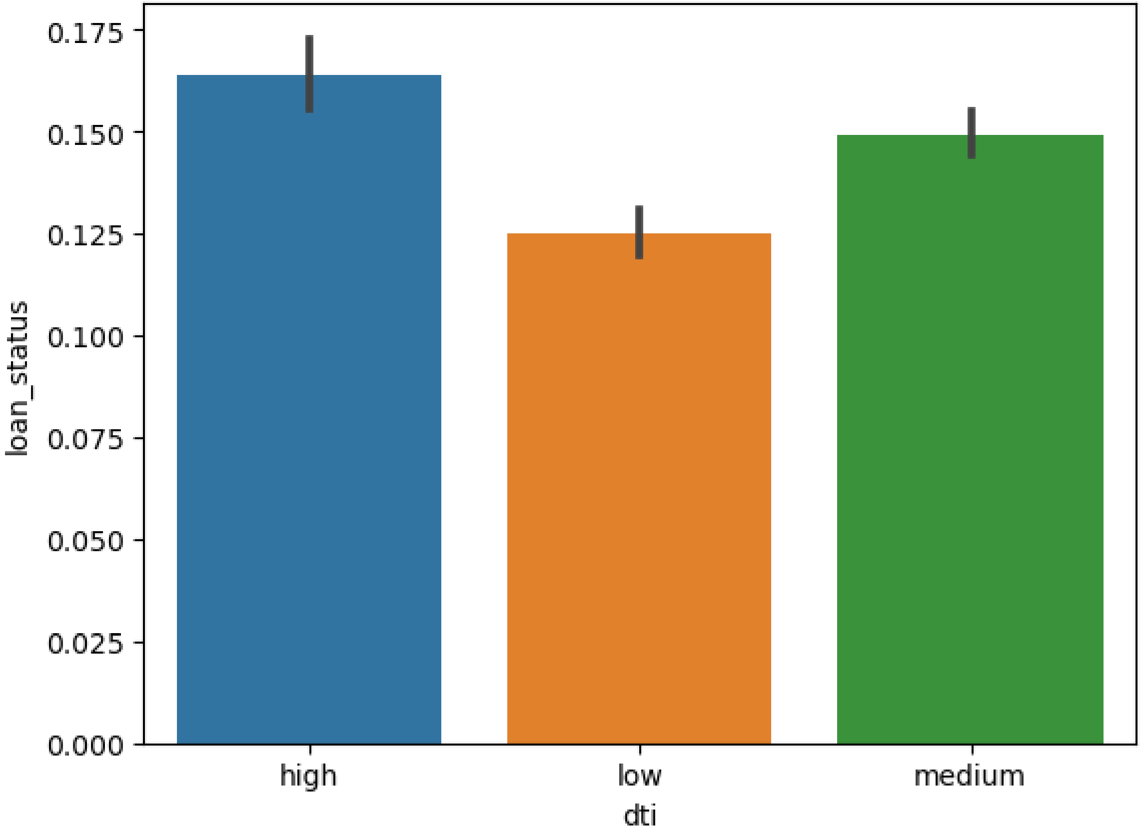
```
def dti(n):
    if n <= 10:
        return 'low'
    elif n > 10 and n <=20:
        return 'medium'
    else:
        return 'high'


loan['dti'] = loan['dti'].apply(lambda x: dti(x))
```

In [234…

```
loan['dti'].value_counts()
```

Out[234]:
```
medium    18002
low       12545
high       6997
Name: dti, dtype: int64
```

In [238…
```
# comparing default rates across debt to income ratio
# higher debt to income ratio has higher default rates
plot_data('dti')
```

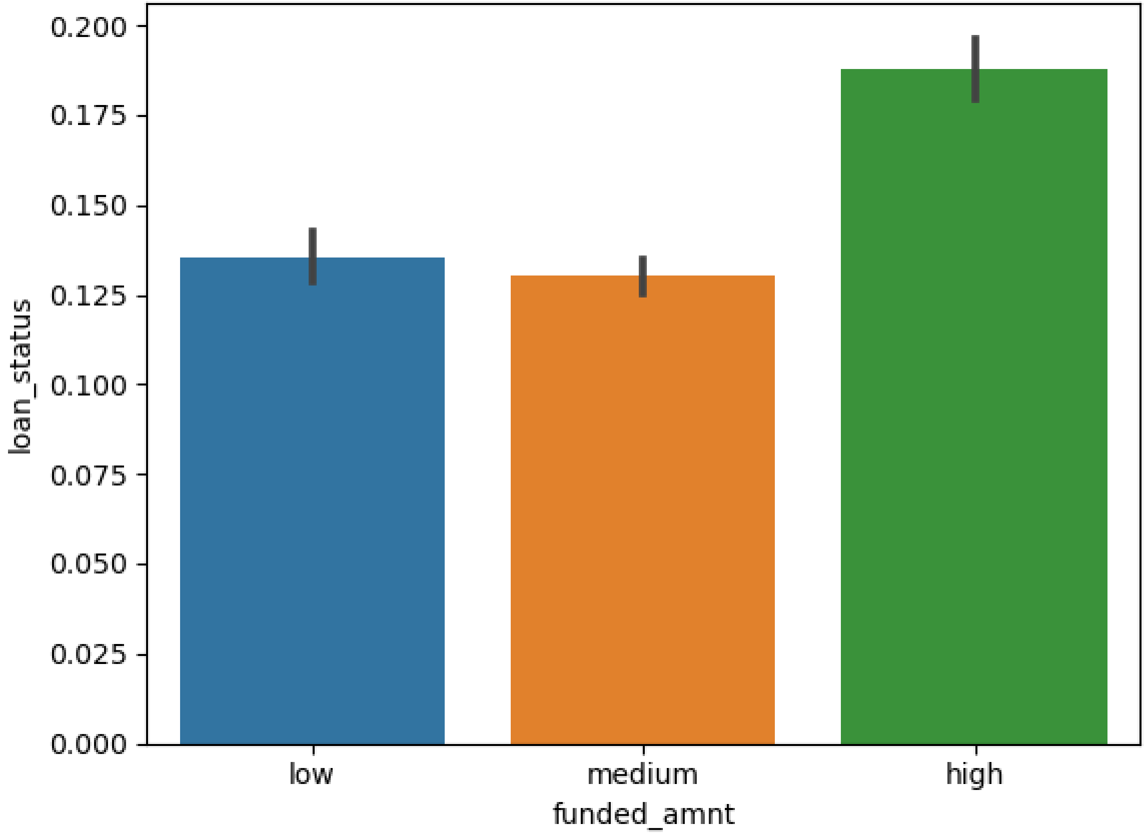


In [241…
```
# funded amount
def funded_amount(n):
    if n <= 5000:
        return 'low'
    elif n > 5000 and n <=15000:
        return 'medium'
    else:
```

```
        return 'high'

loan['funded_amnt'] = loan['funded_amnt'].apply(lambda x: funded_amount(x))
```

In [242…

```
#Higher funds are provided to customer higher is the default rate
plot_data('funded_amnt')
```



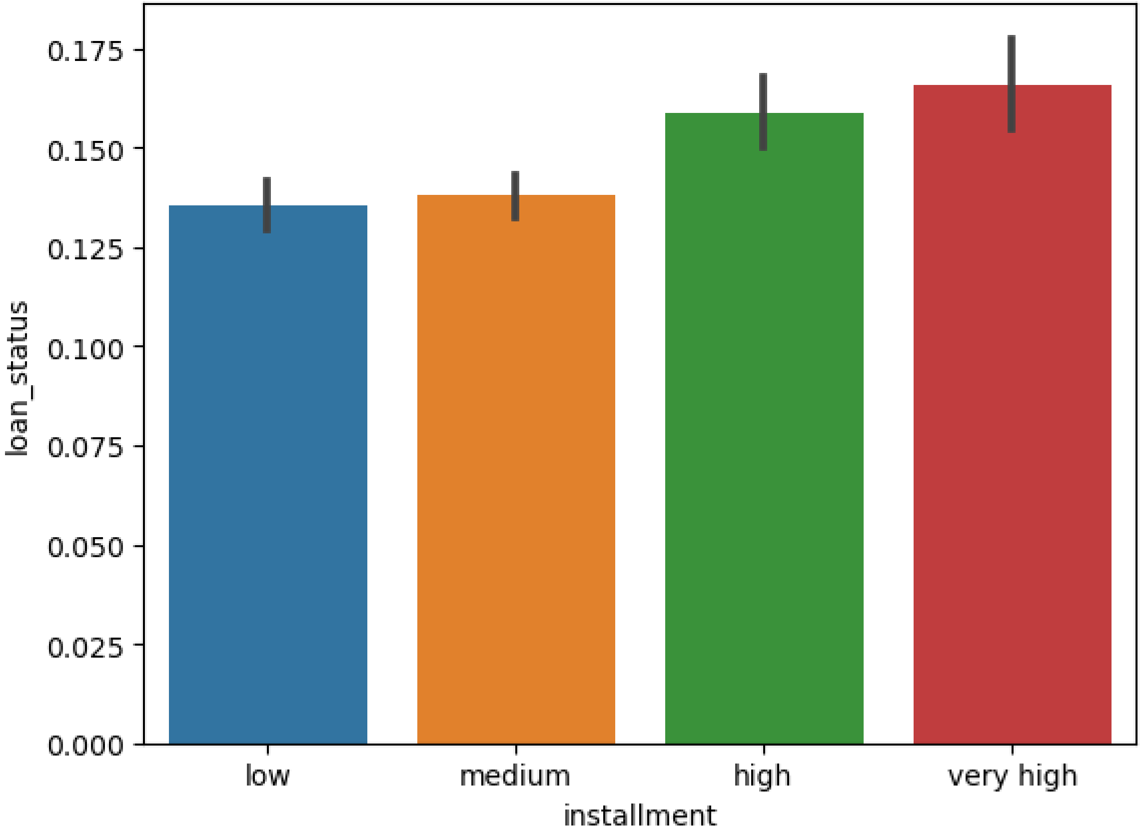
In [243…

```
# installment
def installment(n):
    if n <= 200:
        return 'low'
    elif n > 200 and n <=400:
        return 'medium'
    elif n > 400 and n <=600:
        return 'high'
    else:
```

```
        return 'very high'

loan['installment'] = loan['installment'].apply(lambda x: installment(x))
```

In [244…

```
# the higher the installment amount, the higher the default rate
plot_data('installment')
```



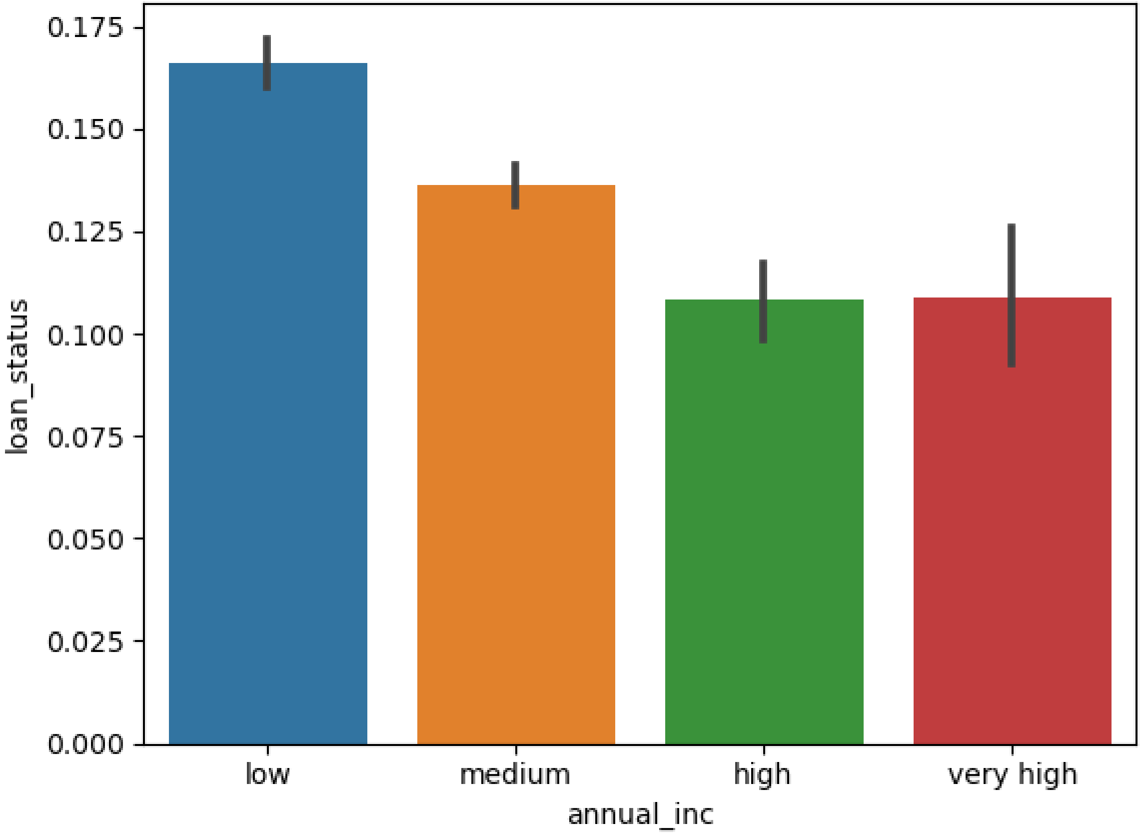
In [245…

```
# annual income
def annual_income(n):
    if n <= 50000:
        return 'low'
    elif n > 50000 and n <=100000:
        return 'medium'
    elif n > 100000 and n <=150000:
        return 'high'
    else:
```

```
        return 'very high'

loan['annual_inc'] = loan['annual_inc'].apply(lambda x: annual_income(x))
```

In [246…

```
#lower the income of customer higher is the default rate
plot_data('annual_inc')
```



In [247…

```
# Emp_length
loan.emp_length.isnull().sum()
```

Out[247]:

```
0
```

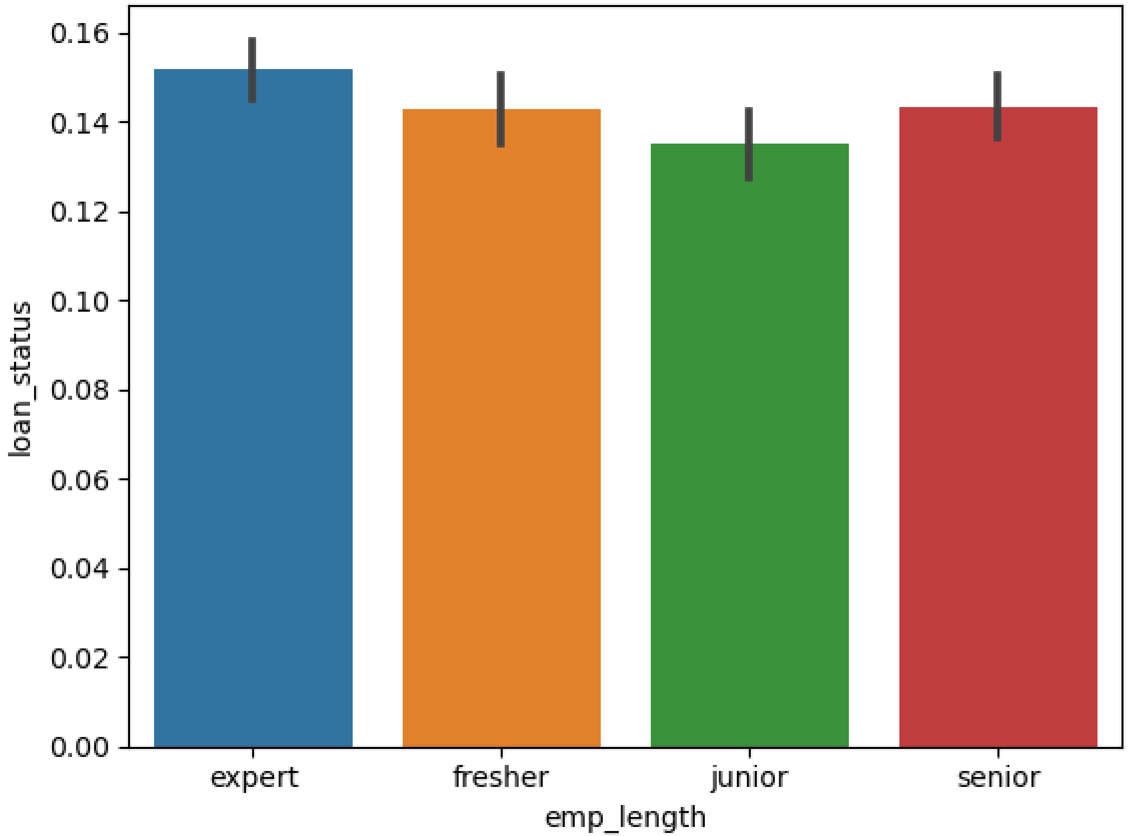
In [249…

```
# binning the variable
def emp_length(n):
    if n <= 1:
```

```
        return 'fresher'
    elif n > 1 and n <=3:
        return 'junior'
    elif n > 3 and n <=7:
        return 'senior'
    else:
        return 'expert'

loan['emp_length'] = loan['emp_length'].apply(lambda x: emp_length(x))
```

In [250…

```
#Suprisingly customers with highest level experience has higher default rate
plot_data('emp_length')
```

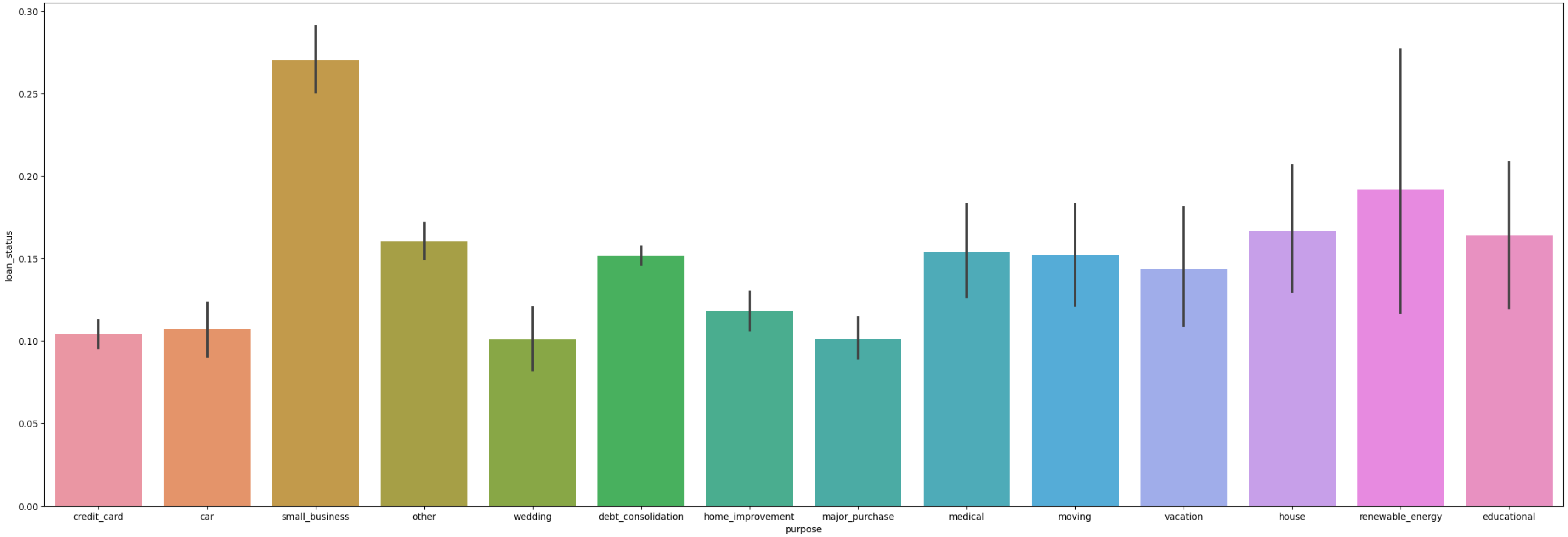


We have now compared the default rates across various variables, and some of the important predictors are purpose of the loan, interest rate, annual income, grade etc.

let's again have a look at the default rates across the purpose of the loan.

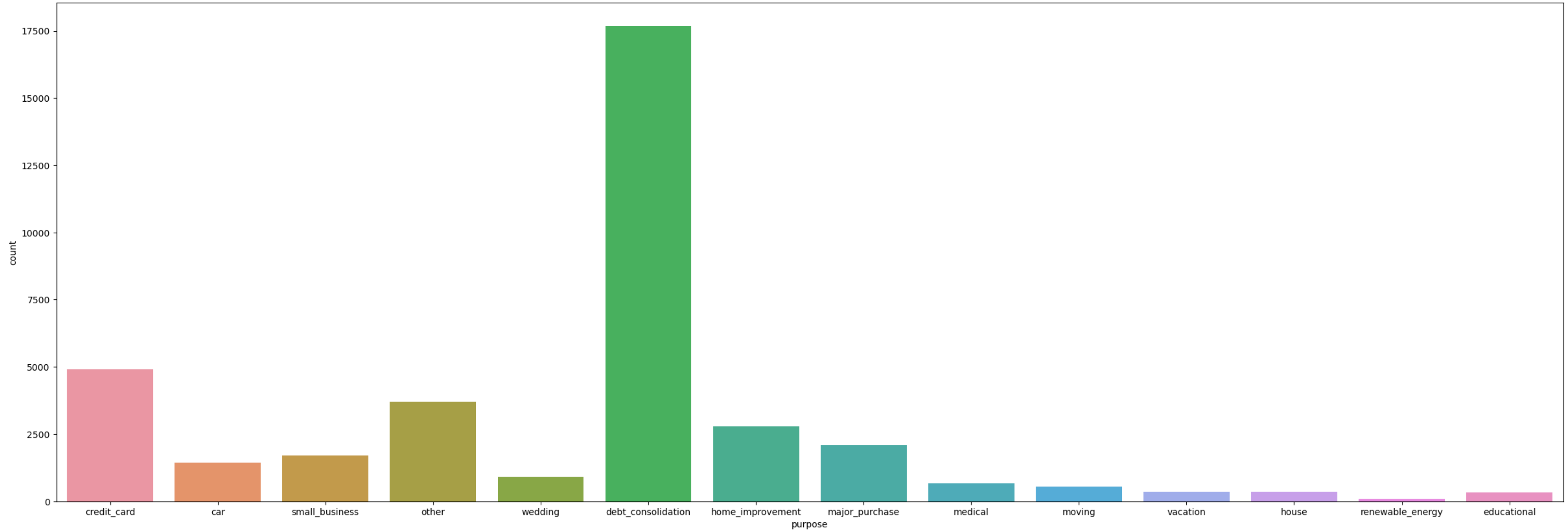
In [255…

```
# purpose: small business loans defualt the most then others
plt.figure(figsize=(30, 10))
plot_data('purpose')
```



In [257…

```
# lets first look at the number of loans for each type (purpose) of the loan
# most loans are debt consolidation(to repay credit card bills or other loan payments), compared to others.
plt.figure(figsize=(30, 10))
sns.countplot(x='purpose', data=loan)
plt.show()
```



In [263…

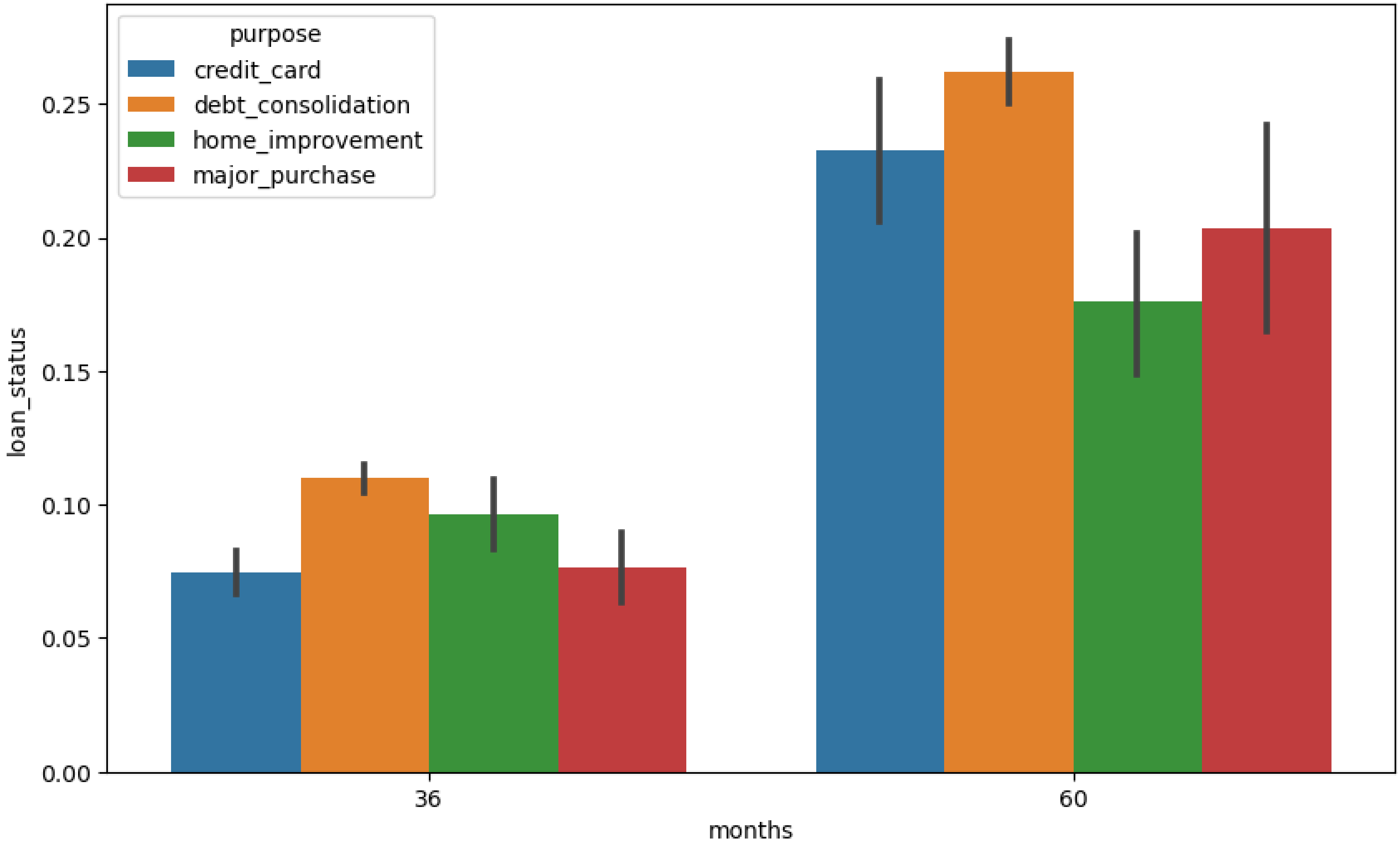
```
# filtering the loan for the 4 types of loans mentioned above
main_purposes = ["credit_card","debt_consolidation","home_improvement","major_purchase"]
loan = loan[loan['purpose'].isin(main_purposes)]
loan['purpose'].value_counts()
```

Out[263]:

```
debt_consolidation    17675
credit_card            4899
home_improvement       2785
major_purchase         2080
Name: purpose, dtype: int64
```

In [265…

```
# let's now compare the default rates across two types of categorical variables
# purpose of loan (constant) and another categorical variable (which changes)

plt.figure(figsize=[10, 6])
sns.barplot(x='months', y="loan_status", hue='purpose', data=loan)
plt.show()
```

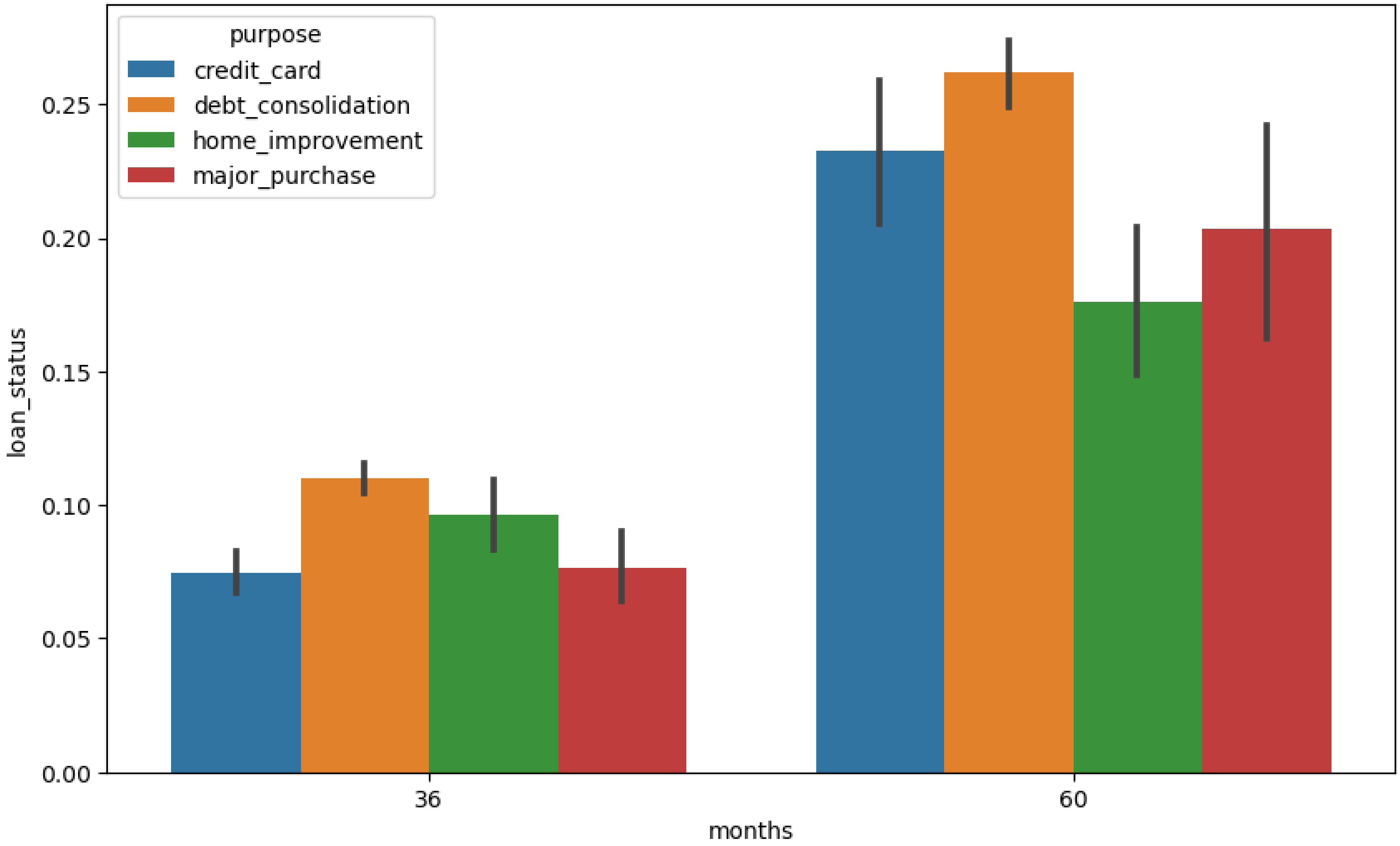


We can clearly see that whether it is 36 months term period or 60 months debt_consolidation has the highest default rate.

In [266…

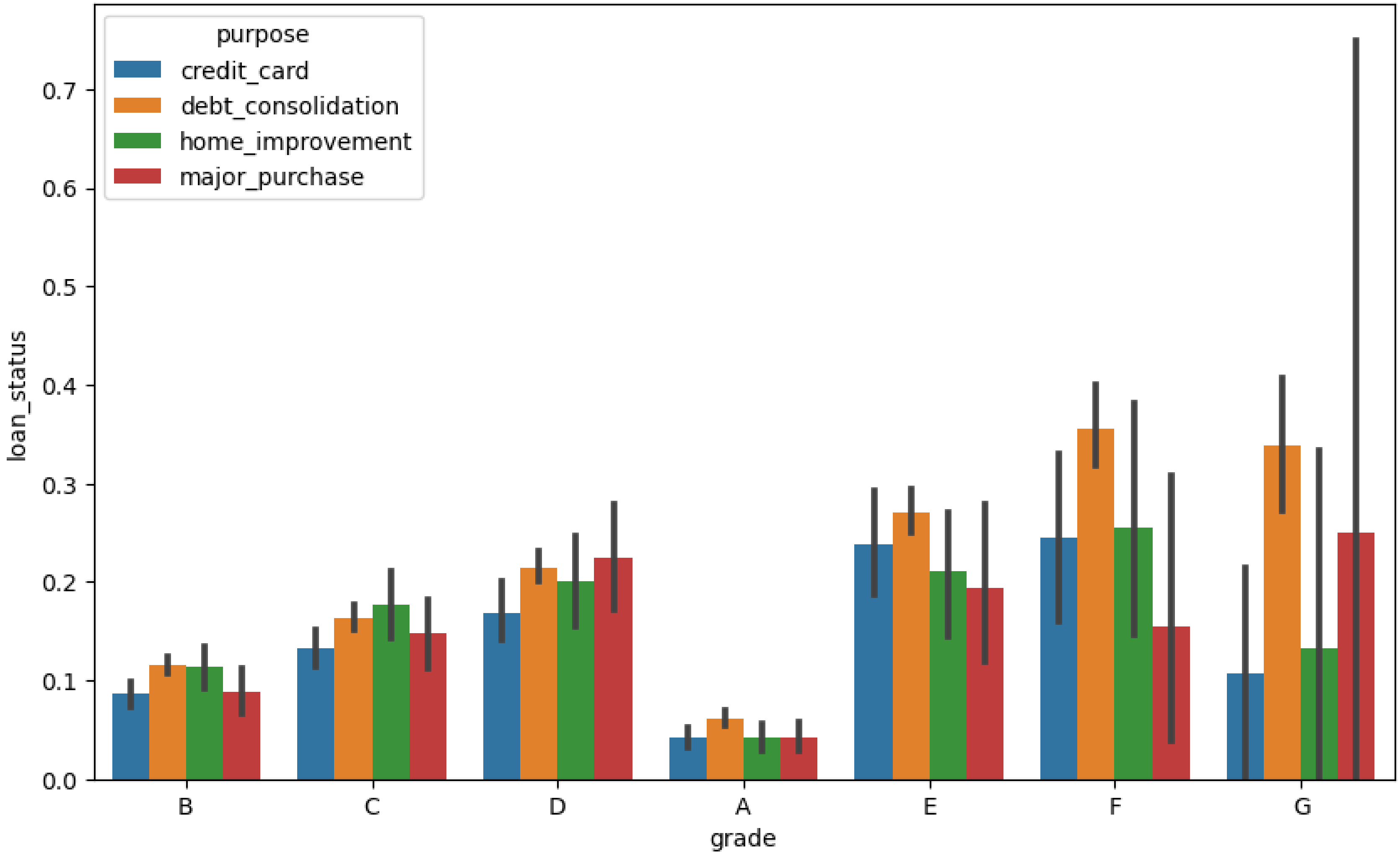
```
def plot_segmented(cat_var):
    plt.figure(figsize=(10, 6))
    sns.barplot(x=cat_var, y='loan_status', hue='purpose', data=loan)
    plt.show()
```

```
plot_segmented('months')
```



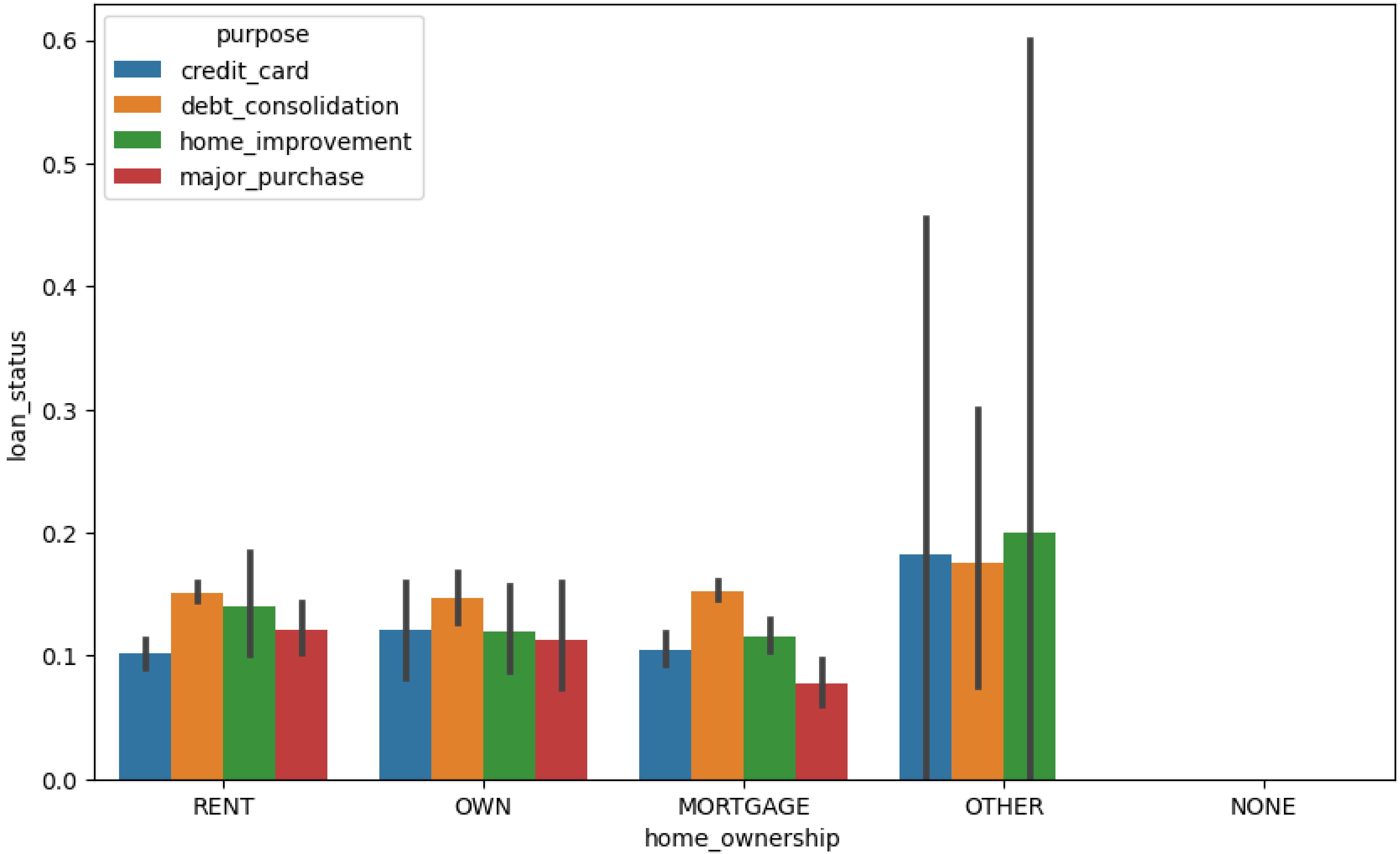
In [267…

```
# grade of loan
plot_segmented('grade')
```



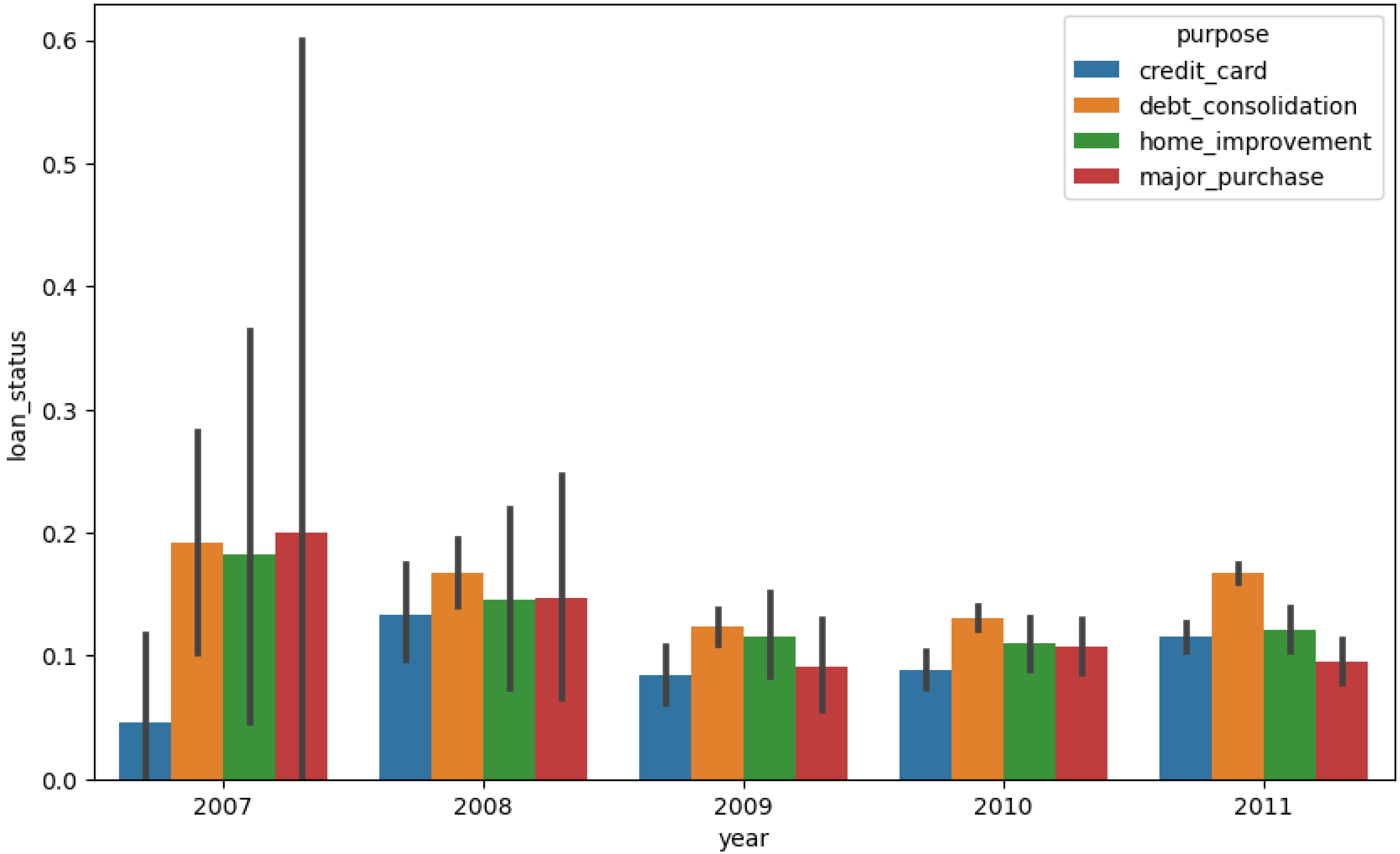
In [268…

```
# home ownership
plot_segmented('home_ownership')
```



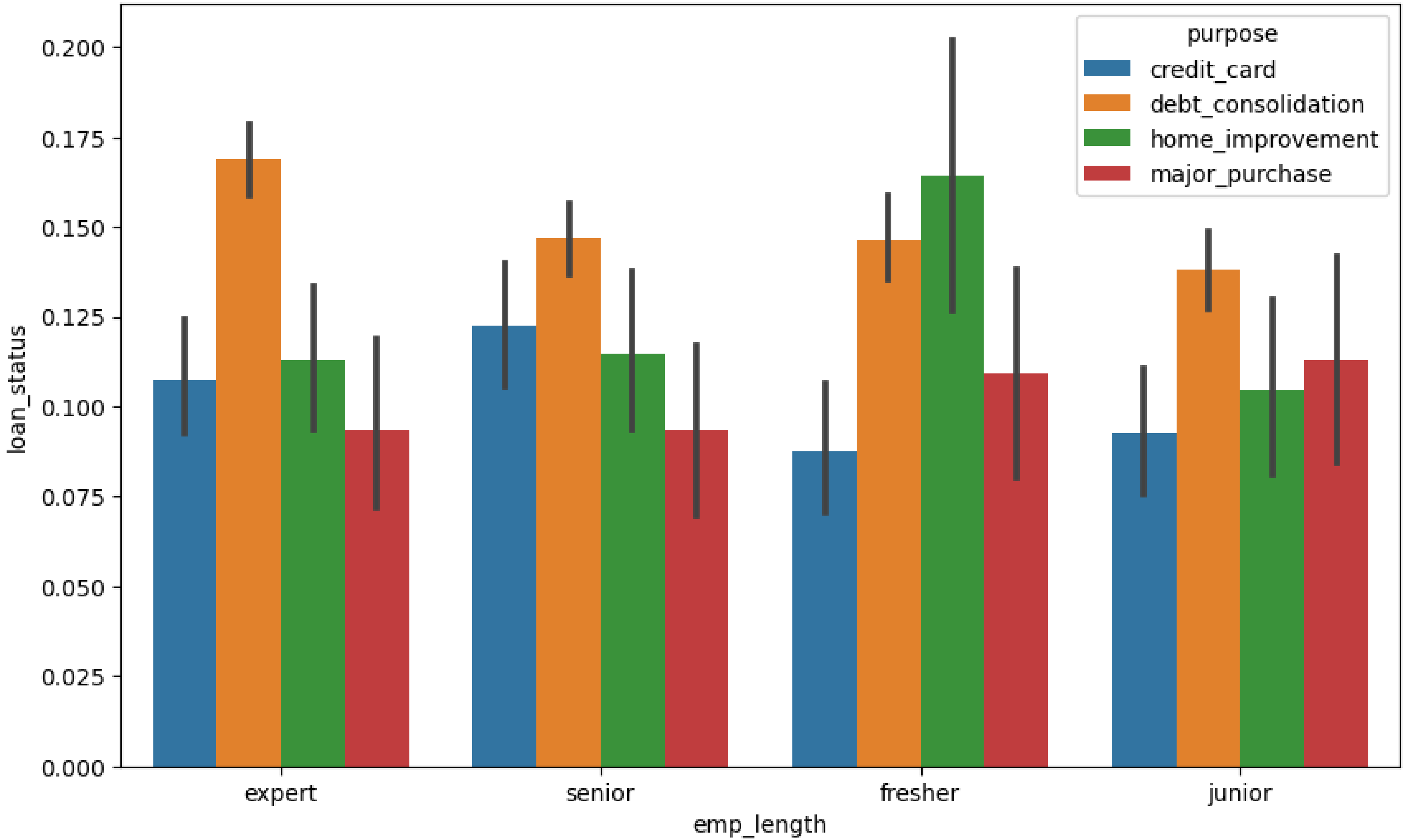
In [269…

```
# year
plot_segmented('year')
```



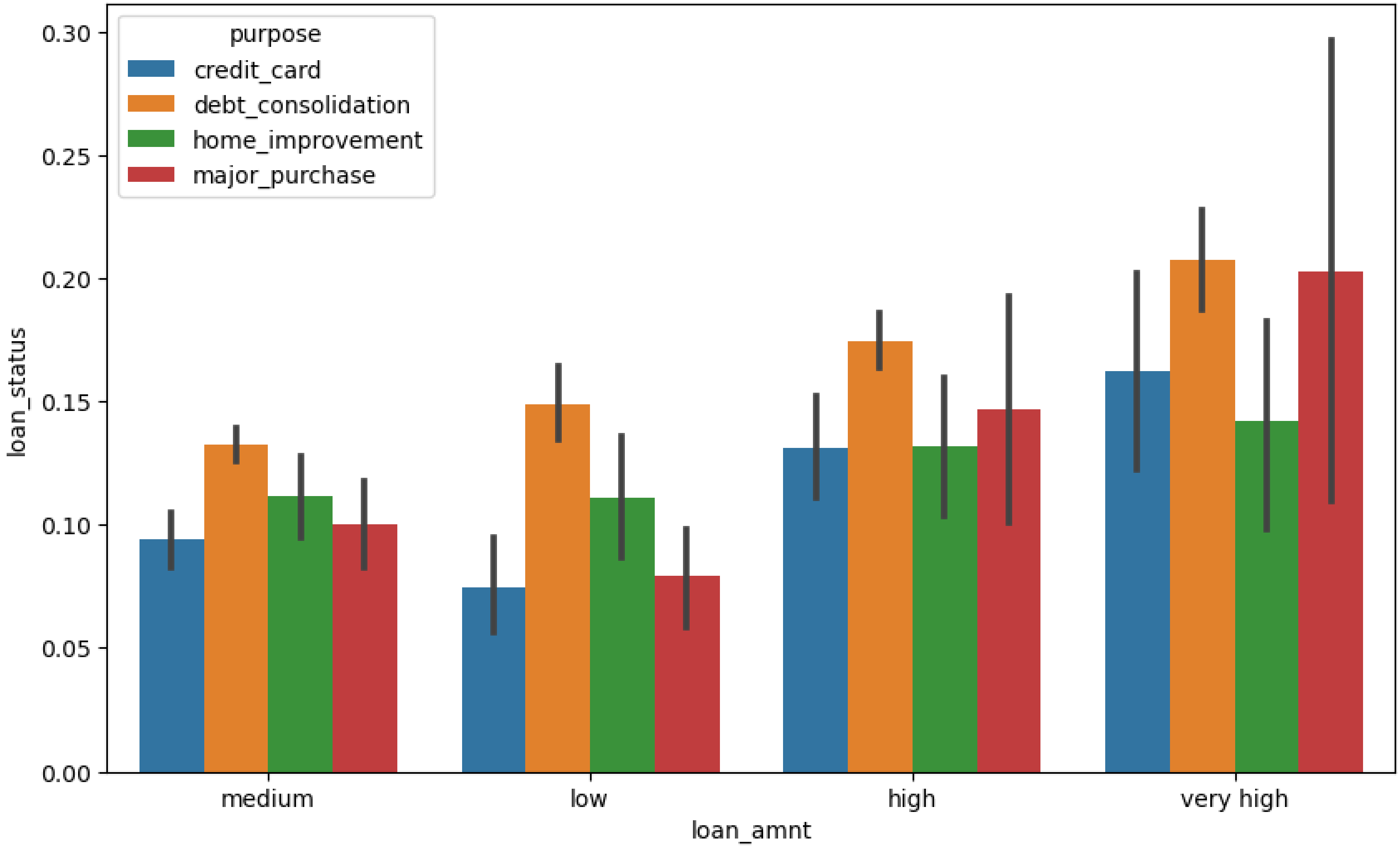
In [270…

```
# emp_length
plot_segmented('emp_length')
```



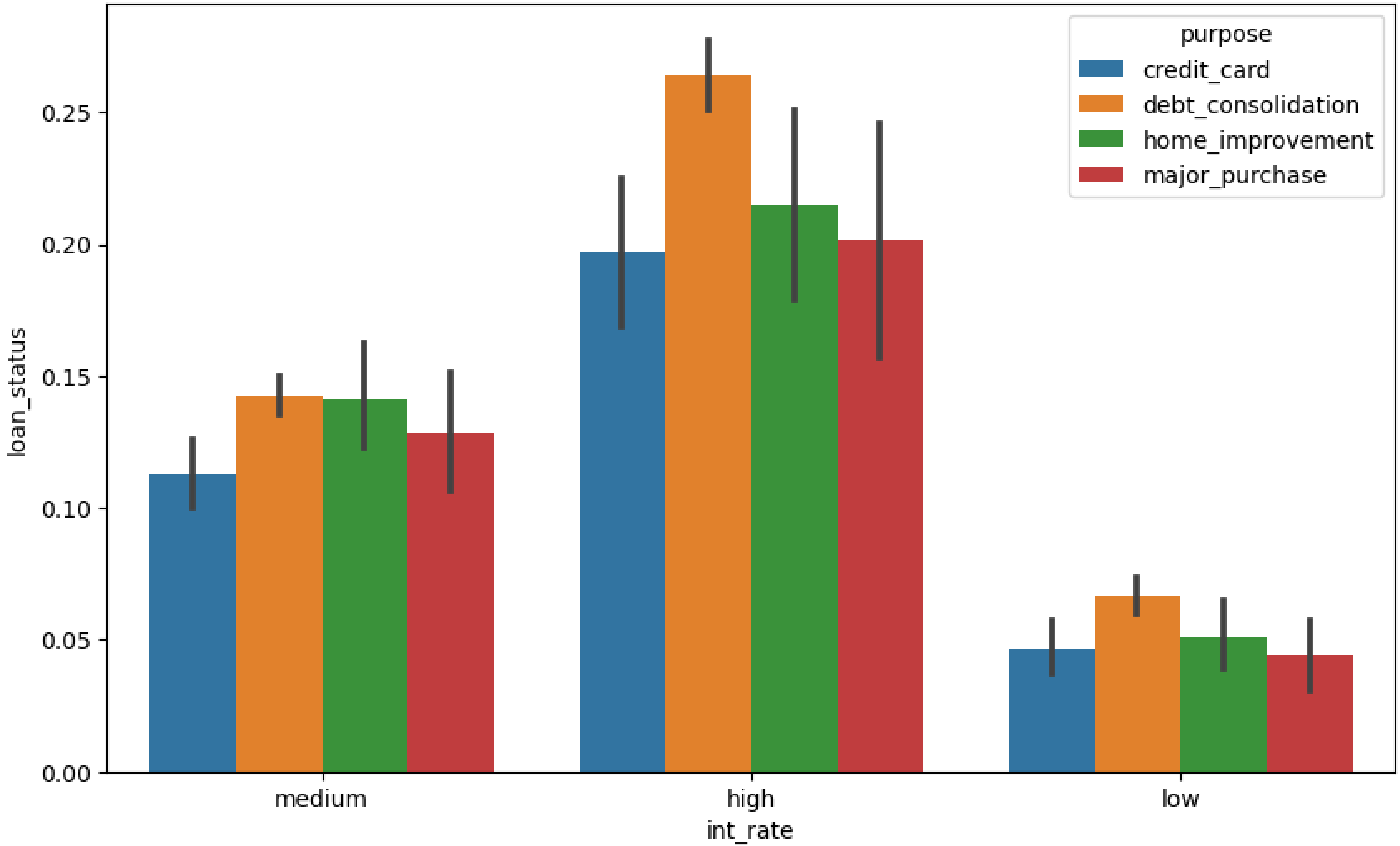
In [271…

```
# loan_amnt: same trend across loan purposes
plot_segmented('loan_amnt')
```



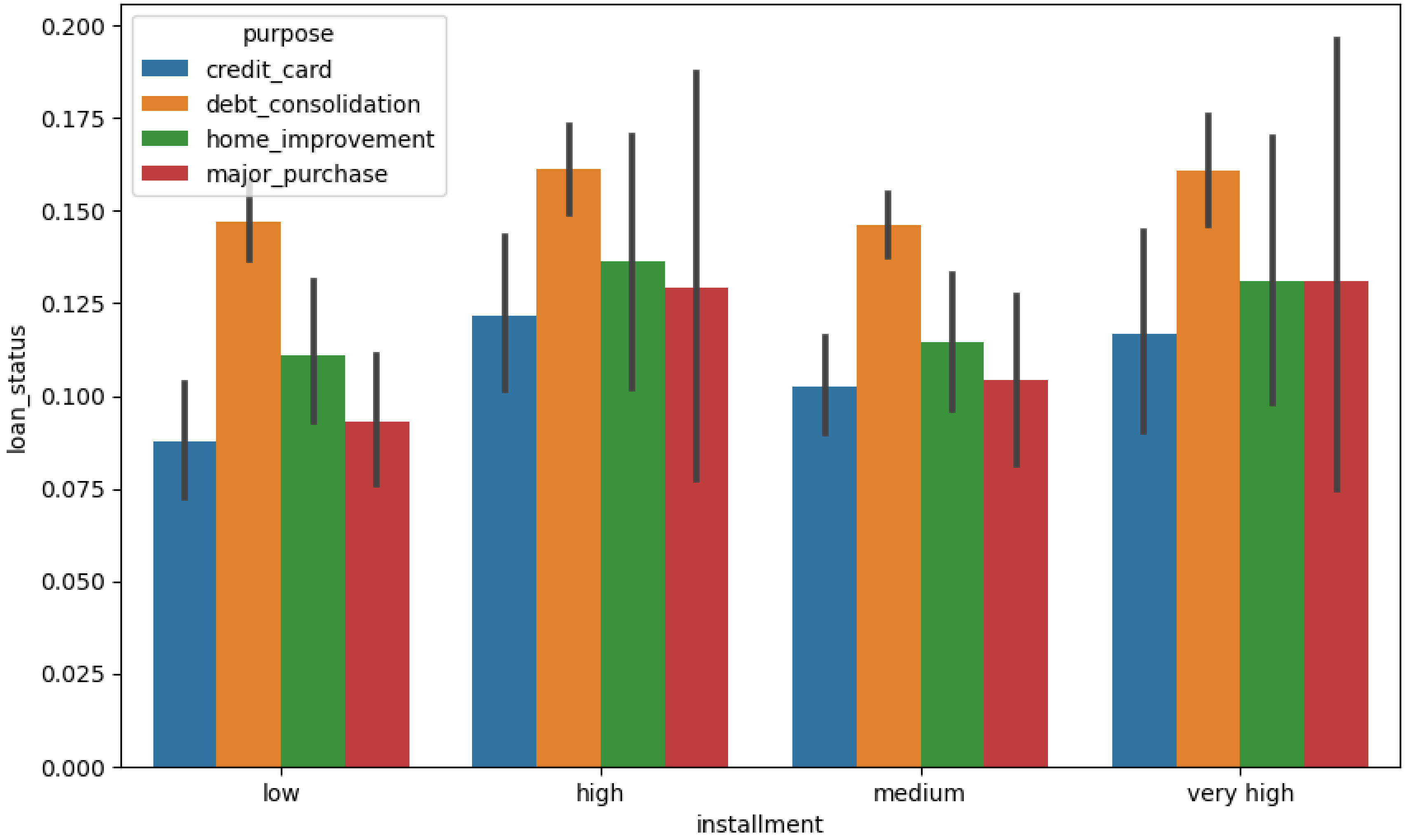
In [272…

```
# interest rate
plot_segmented('int_rate')
```



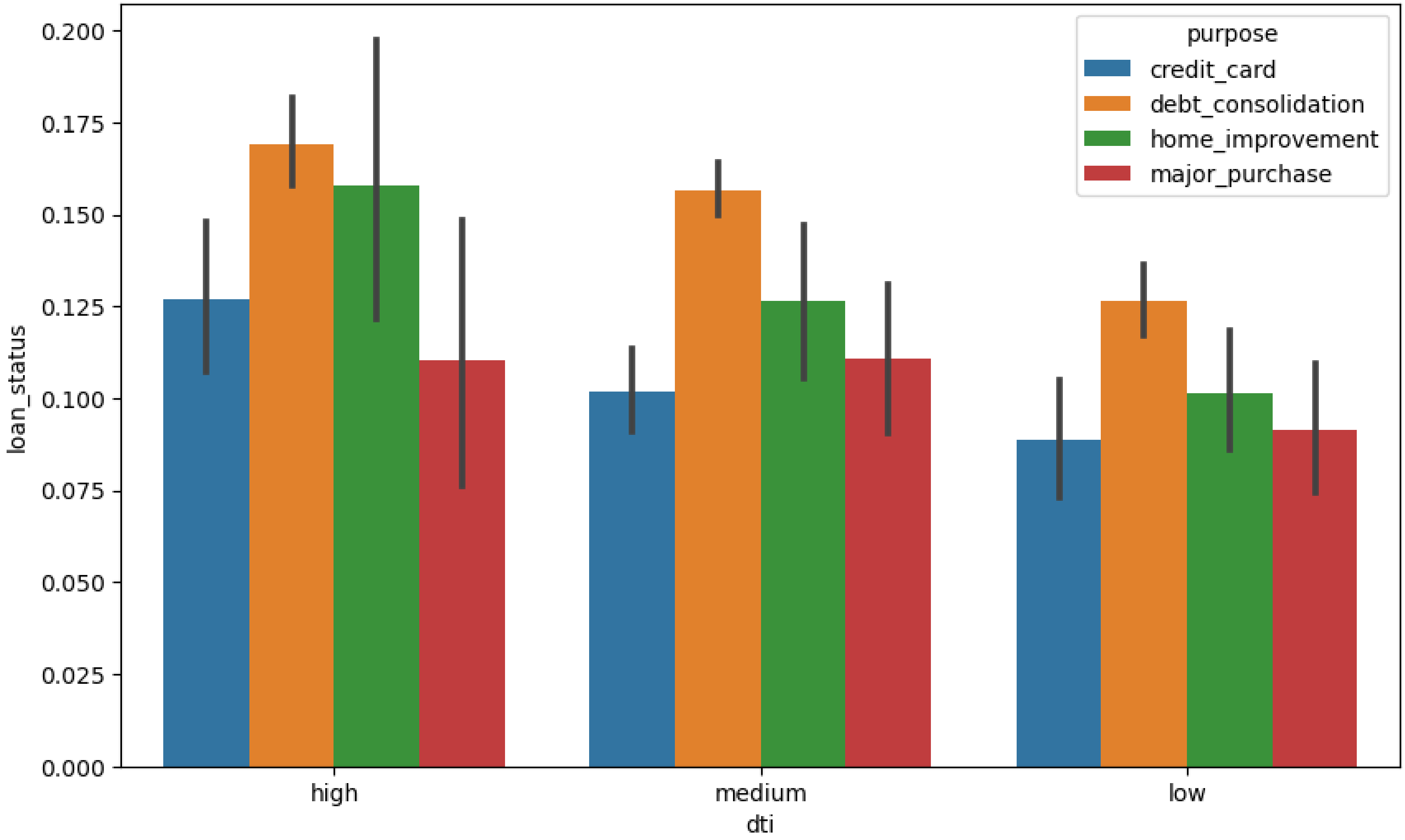
In [273…

```
# installment
plot_segmented('installment')
```



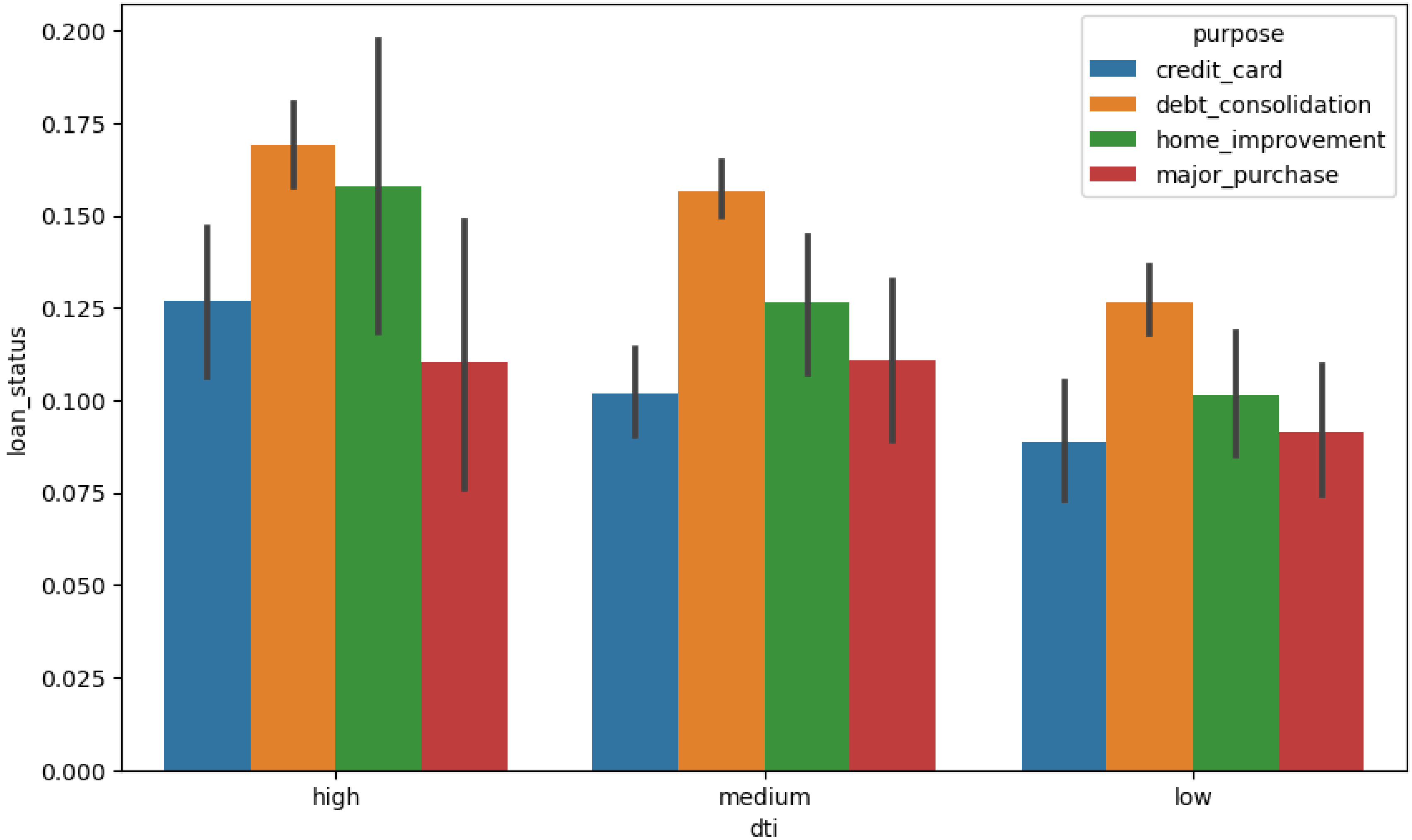
In [274…

```
# debt to income ratio
plot_segmented('dti')
```



In [275…

```
# debt to income ratio
plot_segmented('dti')
```


purpose
credit_card
debt_consolidation
home_improvement
major_purchase
loan_status
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
high
medium
low
dti

```
In [ ]:
```

```
In [ ]:
```