WELCOME TO THE ANALYSIS OF A LENDING CLUB LOAN DATA

1. Importing Relevant libraries and Modules

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import requests
from bs4 import BeautifulSoup
#from datetime import datetime
```

In [2]:
```
#Remove Warnings
import warnings
warnings.filterwarnings("ignore")
```

2.Load your CSV File

In [3]:
```
loan_dataset = pd.read_csv("loan.csv", encoding= "ISO-8859-1") #The encoding is
```

In [4]:
```
#create a shallow copy of this dataset so that our original would be unaffected
loan_dataset_cp = loan_dataset.copy(deep = True)
```

Inspect the data to know what we have

In [5]:
```
loan_dataset_cp.shape
```

Out[5]:
```
(39717, 111)
```

In [6]:
```
loan_dataset_cp.head()
```

Out[6]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	i
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	

5 rows × 111 columns

Dropping Some Irrelevant Columns and Taking only the Ones we need

In [7]:
```
#let's know or let's see the columns we have in our dataset already
```

In [8]:
```
loan_dataset_cp.columns
```

Out[8]:
```
Index(['id', 'member_id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv',
       'term', 'int_rate', 'installment', 'grade', 'sub_grade',
       ...
       'num_tl_90g_dpd_24m', 'num_tl_op_past_12m', 'pct_tl_nvr_dlq',
       'percent_bc_gt_75', 'pub_rec_bankruptcies', 'tax_liens',
       'tot_hi_cred_lim', 'total_bal_ex_mort', 'total_bc_limit',
       'total_il_high_credit_limit'],
      dtype='object', length=111)
```

In [9]:
```
#or rather.., let's store it in an empty list
loan_dataset_columns = [column for column in loan_dataset_cp.columns]
```

In [10]:
```
len(loan_dataset_columns)
```

Out[10]:
```
111
```

In [11]:
```
for column in loan_dataset_columns:
    print(column)
```

```
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
mths_since_last_major_derog
policy_code
application_type
annual_inc_joint
dti_joint
verification_status_joint
acc_now_delinq
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
chargeoff_within_12_mths
delinq_amnt
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
pub_rec_bankruptcies
tax_liens
tot_hi_cred_lim
total_bal_ex_mort
total_bc_limit
total_il_high_credit_limit
```

In [12]:
```
'acc_now_delinq' in loan_dataset_columns
```

Out[12]: True

In [13]:
```
loan_dataset_columns.sort()
```

In [14]:
```
for columns in loan_dataset_columns:
    print(columns)
```

```
acc_now_delinq
acc_open_past_24mths
addr_state
all_util
annual_inc
annual_inc_joint
application_type
avg_cur_bal
bc_open_to_buy
bc_util
chargeoff_within_12_mths
collection_recovery_fee
collections_12_mths_ex_med
delinq_2yrs
delinq_amnt
desc
dti
dti_joint
earliest_cr_line
emp_length
emp_title
funded_amnt
funded_amnt_inv
grade
home_ownership
id
il_util
initial_list_status
inq_fi
inq_last_12m
inq_last_6mths
installment
int_rate
issue_d
last_credit_pull_d
last_pymnt_amnt
last_pymnt_d
loan_amnt
loan_status
max_bal_bc
member_id
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_last_delinq
mths_since_last_major_derog
mths_since_last_record
mths_since_rcnt_il
mths_since_recent_bc
mths_since_recent_bc_dlq
mths_since_recent_inq
mths_since_recent_revol_delinq
next_pymnt_d
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
open_acc
open_acc_6m
open_il_12m
open_il_24m
open_il_6m
open_rv_12m
open_rv_24m
out_prncp
out_prncp_inv
pct_tl_nvr_dlq
percent_bc_gt_75
policy_code
pub_rec
pub_rec_bankruptcies
purpose
pymnt_plan
recoveries
revol_bal
revol_util
sub_grade
tax_liens
term
title
tot_coll_amt
tot_cur_bal
tot_hi_cred_lim
total_acc
total_bal_ex_mort
total_bal_il
total_bc_limit
total_cu_tl
total_il_high_credit_limit
total_pymnt
total_pymnt_inv
total_rec_int
total_rec_late_fee
total_rec_prncp
total_rev_hi_lim
url
verification_status
verification_status_joint
zip_code
```

In [15]:
```
#Rather than dropping, which is more tedious let's carry the ones we need
```

In [16]:
```
#let's just drop regardless
```

In [17]:
```
loan_dataset_cp = loan_dataset_cp.drop([
'acc_now_delinq'
,'acc_open_past_24mths'
```

```
,'all_util'
,'annual_inc_joint'
,'application_type'
,'avg_cur_bal'
,'bc_open_to_buy'
,'bc_util'
,'chargeoff_within_12_mths'
,'collection_recovery_fee'
,'collections_12_mths_ex_med'
,'delinq_2yrs'
,'delinq_amnt'
,'desc'
,'dti_joint'
,'earliest_cr_line'
,'home_ownership'
,'id'
,'il_util'
,'initial_list_status'
,'inq_fi'
,'inq_last_12m'
,'max_bal_bc'
,'member_id'
,'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_last_delinq'
,'mths_since_last_major_derog'
,'mths_since_last_record'
,'mths_since_rcnt_il'
,'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'
,'open_acc_6m'
,'open_il_12m'
,'open_il_24m'
,'open_il_6m'
,'open_rv_12m'
,'open_rv_24m'
,'out_prncp'
,'out_prncp_inv'
,'pct_tl_nvr_dlq'
,'percent_bc_gt_75'
,'policy_code'
```

```
,'pymnt_plan'
,'recoveries'
,'tax_liens'
,'title'
,'tot_coll_amt'
,'tot_cur_bal'
,'tot_hi_cred_lim'
,'total_bal_ex_mort'
,'total_bal_il'
,'total_bc_limit'
,'total_cu_tl'
,'next_pymnt_d'
,'total_il_high_credit_limit'
,'total_pymnt'
,'total_pymnt_inv'
,'total_rec_int'
,'total_rec_late_fee'
,'total_rec_prncp'
,'verification_status_joint'
,'total_rev_hi_lim'
,'last_pymnt_amnt'
,'last_credit_pull_d'
,'url'], axis='columns')
```

In [18]:
```
loan_dataset_cp.shape
```

Out[18]: (39717, 26)

In [19]:
```
loan_dataset_cp.head()
```

Out[19]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	su
0	5000	5000	4975.0	36 months	10.65%	162.87	B	
1	2500	2500	2500.0	60 months	15.27%	59.83	C	
2	2400	2400	2400.0	36 months	15.96%	84.33	C	
3	10000	10000	10000.0	36 months	13.49%	339.31	C	
4	3000	3000	3000.0	60 months	12.69%	67.79	B	

5 rows × 26 columns

In [20]:
```
#let's gain more insight into our data
```

In [21]:
```
loan_dataset_cp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39717 entries, 0 to 39716
Data columns (total 26 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   loan_amnt             39717 non-null  int64  
 1   funded_amnt           39717 non-null  int64  
 2   funded_amnt_inv       39717 non-null  float64
 3   term                  39717 non-null  object 
 4   int_rate              39717 non-null  object 
 5   installment           39717 non-null  float64
 6   grade                 39717 non-null  object 
 7   sub_grade             39717 non-null  object 
 8   emp_title             37258 non-null  object 
 9   emp_length            38642 non-null  object 
 10  annual_inc            39717 non-null  float64
 11  verification_status   39717 non-null  object 
 12  issue_d               39717 non-null  object 
 13  loan_status           39717 non-null  object 
 14  purpose               39717 non-null  object 
 15  zip_code              39717 non-null  object 
 16  addr_state            39717 non-null  object 
 17  dti                   39717 non-null  float64
 18  inq_last_6mths        39717 non-null  int64  
 19  open_acc              39717 non-null  int64  
 20  pub_rec               39717 non-null  int64  
 21  revol_bal             39717 non-null  int64  
 22  revol_util            39667 non-null  object 
 23  total_acc             39717 non-null  int64  
 24  last_pymnt_d          39646 non-null  object 
 25  pub_rec_bankruptcies  39020 non-null  float64
dtypes: float64(5), int64(7), object(14)
memory usage: 7.9+ MB
```

In [22]:
```
#let's take a look at the memory usage alright
```

In [23]:
```
loan_dataset_cp.memory_usage()
```

Out[23]:
```
Index                      132
loan_amnt               317736
funded_amnt             317736
funded_amnt_inv         317736
term                    317736
int_rate                317736
installment             317736
grade                   317736
sub_grade               317736
emp_title               317736
emp_length              317736
annual_inc              317736
verification_status     317736
issue_d                 317736
loan_status             317736
purpose                 317736
zip_code                317736
addr_state              317736
dti                     317736
inq_last_6mths          317736
open_acc                317736
pub_rec                 317736
revol_bal               317736
revol_util              317736
total_acc               317736
last_pymnt_d            317736
pub_rec_bankruptcies    317736
dtype: int64
```

In [24]:
```
type(loan_dataset_cp.memory_usage())
```

Out[24]:
```
pandas.core.series.Series
```

In [25]:
```
317736/1000
```

Out[25]:
```
317.736
```

In [26]:
```
317/1000 * 26
```

Out[26]:
```
8.242
```

Let's do a quick transformation using Apply/Map - Lambda

This is Actually a little bit of feature Engineering

In [27]:
```
loan_dataset_cp.loan_status.head()
```

Out[27]:
```
0      Fully Paid
1     Charged Off
2      Fully Paid
3      Fully Paid
4         Current
Name: loan_status, dtype: object
```

In [28]:
```
#let's create a new column called defaulted that returns True(1) if it was charg
#and False (0) if it was fully paid
```

In [29]:
```
loan_dataset_cp['defaulted'] = loan_dataset_cp['loan_status'].map(lambda x: 1 if
#this could also be like this alright
#loan_dataset_cp['defaulted'] = loan_dataset_cp['loan_status'].apply(lambda x: 1
```

In [30]:
```
#observe, we can't see all the columns, let's set the pd options of the column s
```

In [31]:
```
pd.options.display.max_columns = 30
```

In [32]:
```
loan_dataset_cp.head()
```

Out[32]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	su
0	5000	5000	4975.0	36 months	10.65%	162.87	B	
1	2500	2500	2500.0	60 months	15.27%	59.83	C	
2	2400	2400	2400.0	36 months	15.96%	84.33	C	
3	10000	10000	10000.0	36 months	13.49%	339.31	C	
4	3000	3000	3000.0	60 months	12.69%	67.79	B	

In [33]:
```
#Let's see some statistical analysis of this data
```

In [34]:
```
loan_dataset_cp.describe()
```

Out[34]:

	loan_amnt	funded_amnt	funded_amnt_inv	installment	annual_inc	
count	39717.000000	39717.000000	39717.000000	39717.000000	3.971700e+04	3971
mean	11219.443815	10947.713196	10397.448868	324.561922	6.896893e+04	1.
std	7456.670694	7187.238670	7128.450439	208.874874	6.379377e+04	
min	500.000000	500.000000	0.000000	15.690000	4.000000e+03	
25%	5500.000000	5400.000000	5000.000000	167.020000	4.040400e+04	
50%	10000.000000	9600.000000	8975.000000	280.220000	5.900000e+04	1.
75%	15000.000000	15000.000000	14400.000000	430.780000	8.230000e+04	1.
max	35000.000000	35000.000000	35000.000000	1305.190000	6.000000e+06	2

In [35]:
```
type(loan_dataset_cp.describe())
```

Out[35]:
```
pandas.core.frame.DataFrame
```

In [36]:
```
loan_dataset_cp.describe().loan_amnt
```

Out[36]:
```
count    39717.000000
mean     11219.443815
std       7456.670694
min        500.000000
25%       5500.000000
50%      10000.000000
75%      15000.000000
max      35000.000000
Name: loan_amnt, dtype: float64
```

In [37]:
```
#general information of the dataset
```

In [38]:
```
loan_dataset_cp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39717 entries, 0 to 39716
Data columns (total 27 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   loan_amnt             39717 non-null  int64  
 1   funded_amnt           39717 non-null  int64  
 2   funded_amnt_inv       39717 non-null  float64
 3   term                  39717 non-null  object 
 4   int_rate              39717 non-null  object 
 5   installment           39717 non-null  float64
 6   grade                 39717 non-null  object 
 7   sub_grade             39717 non-null  object 
 8   emp_title             37258 non-null  object 
 9   emp_length            38642 non-null  object 
 10  annual_inc            39717 non-null  float64
 11  verification_status   39717 non-null  object 
 12  issue_d               39717 non-null  object 
 13  loan_status           39717 non-null  object 
 14  purpose               39717 non-null  object 
 15  zip_code              39717 non-null  object 
 16  addr_state            39717 non-null  object 
 17  dti                   39717 non-null  float64
 18  inq_last_6mths        39717 non-null  int64  
 19  open_acc              39717 non-null  int64  
 20  pub_rec               39717 non-null  int64  
 21  revol_bal             39717 non-null  int64  
 22  revol_util            39667 non-null  object 
 23  total_acc             39717 non-null  int64  
 24  last_pymnt_d          39646 non-null  object 
 25  pub_rec_bankruptcies  39020 non-null  float64
 26  defaulted             39717 non-null  int64  
dtypes: float64(5), int64(8), object(14)
memory usage: 8.2+ MB
```

UNIVARIATE ANALYSIS

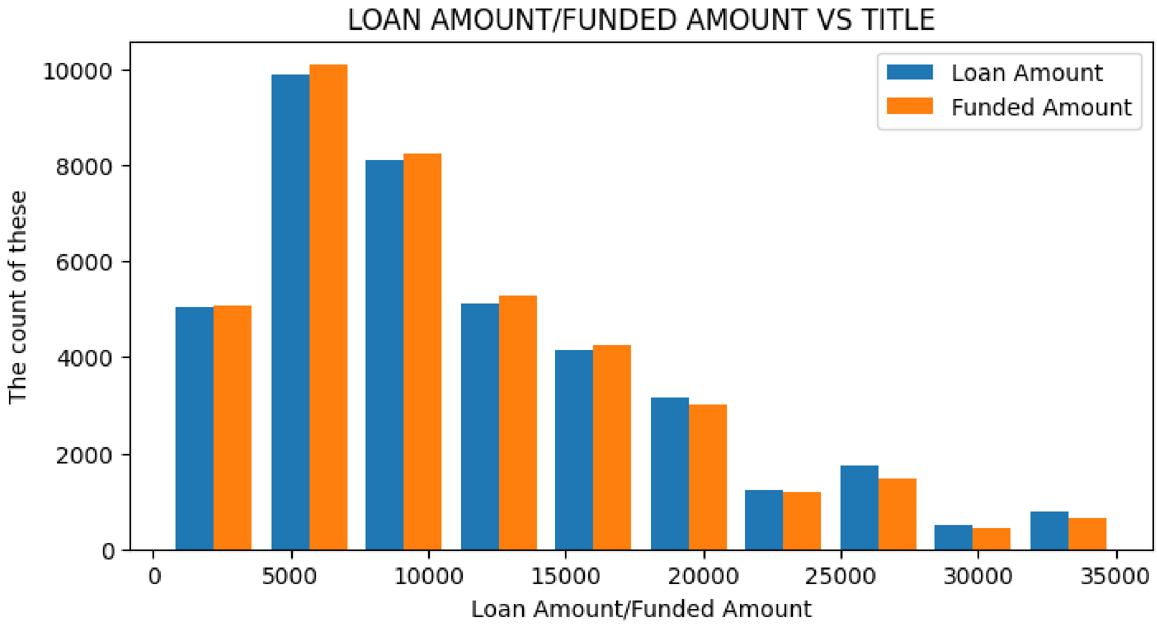
In [39]:
```
#Aim: To checkout the distribution of Loan Amounts and Funded Amounts
```

In [40]:
```
for values in loan_dataset_cp.columns:
    if values.endswith('amnt'):
        print(values)
```

```
    else:
        pass
```

```
loan_amnt
funded_amnt
```

In [41]:
```
fig = plt.figure(figsize = (8,4))
plt.hist(x = [loan_dataset_cp.loan_amnt, loan_dataset_cp.funded_amnt], label = [
plt.xlabel("Loan Amount/Funded Amount")
plt.ylabel("The count of these")
plt.title("LOAN AMOUNT/FUNDED AMOUNT VS TITLE")
plt.legend()
fig.show()
```



Just a pivot_table analysis

In [42]:
```
#let's do average funded amount for defaulters and not defauotrers for the 36 an
```

In [43]:
```
loan_dataset_cp.columns
```

Out[43]:
```
Index(['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'term', 'int_rate',
       'installment', 'grade', 'sub_grade', 'emp_title', 'emp_length',
       'annual_inc', 'verification_status', 'issue_d', 'loan_status',
       'purpose', 'zip_code', 'addr_state', 'dti', 'inq_last_6mths',
       'open_acc', 'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
       'last_pymnt_d', 'pub_rec_bankruptcies', 'defaulted'],
      dtype='object')
```

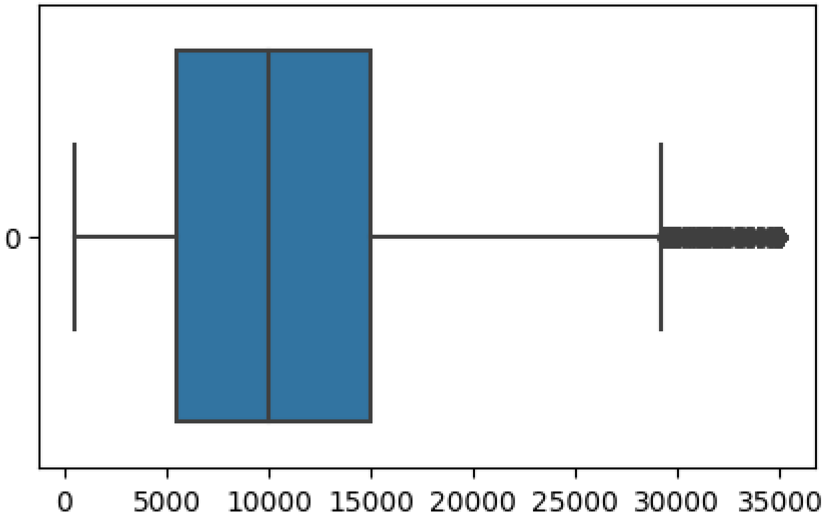
In [44]:
```
loan_dataset_cp.pivot_table(index = 'defaulted', columns = 'term', values = 'fun
```

Out[44]:

term	36 months	60 months
defaulted		
0	9495.432757	14966.135507
1	9258.064766	15108.583333

In [45]:
```
#let's get a visual representation or understanding of the amounnt of loans that
```

In [46]:
```
plt.figure(figsize = (5,3))
sns.boxplot(loan_dataset_cp.loan_amnt, orient = 'horizontal')
plt.show()
```



In [47]:
```
#let's examine the income of the defaulters and the non defaulter.., we could us
#just to see probably the average
```

In [48]:
```
loan_dataset_cp.head(1)
```

Out[48]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	su
0	5000	5000	4975.0	36 months	10.65%	162.87	B	

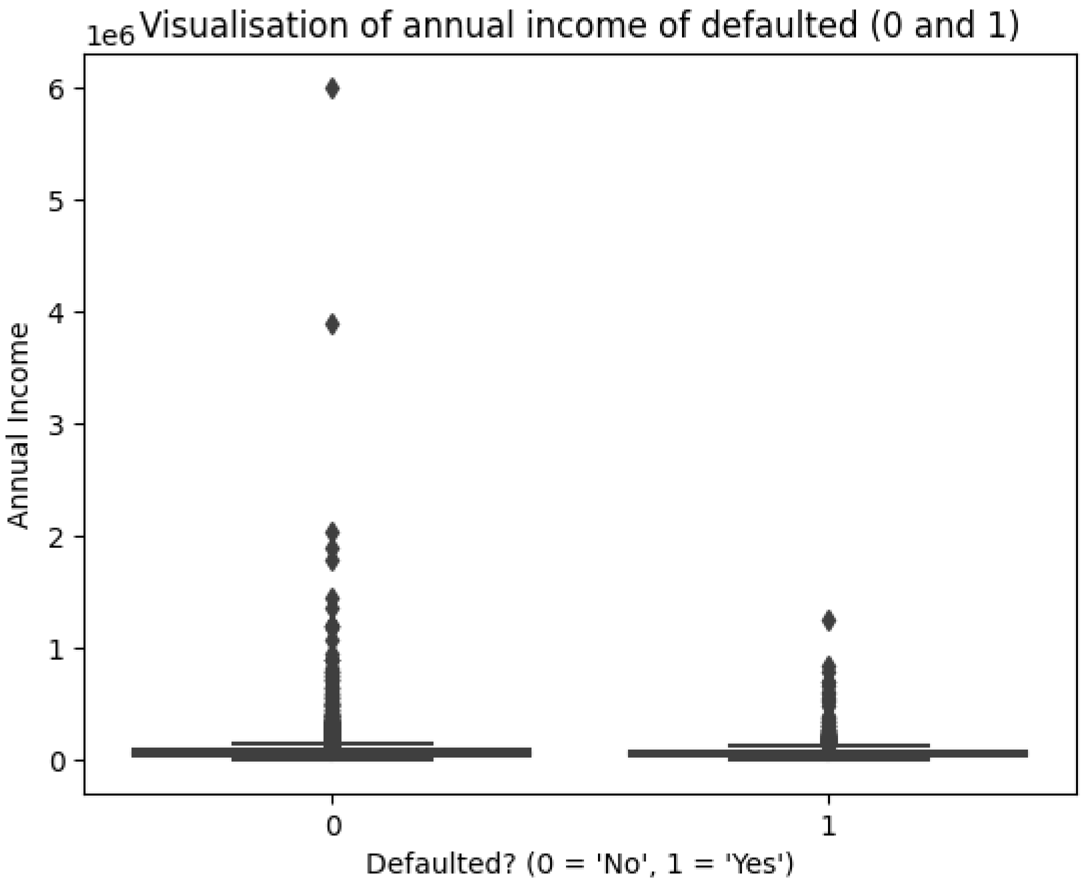
In [49]:
```
loan_dataset_cp.pivot_table(index = 'defaulted', values = 'annual_inc', aggfunc
```

Out[49]:

	annual_inc
defaulted	
0	70048.707623
1	62427.298034

In [50]:
```
fig, ax = plt.subplots()
ax = sns.boxplot(x = 'defaulted', y = 'annual_inc', data = loan_dataset_cp)
ax.set_xlabel("Defaulted? (0 = 'No', 1 = 'Yes')")
ax.set_ylabel("Annual Income")
ax.set_title("Visualisation of annual income of defaulted (0 and 1)")
fig.show()

# fig = plt.figure()
# ax = fig.add_subplot(1,1,1)
# ax = sns.boxplot(x = 'defaulted', y = 'annual_inc', data = loan_dataset_cp)
# ax.set_xlabel("Defaulted? (0 = 'No', 1 = 'Yes')")
# ax.set_ylabel("Annual Income")
# ax.set_title("Visualisation of annual income of defaulted (0 and 1)")
# fig.show()
```

```
# plt.figure()
# sns.boxplot(x = 'defaulted', y = 'annual_inc', data = loan_dataset_cp)
# plt.xlabel("Defaulted? (0 = 'No', 1 = 'Yes')")
# plt.ylabel("Annual Income")
# plt.title("Visualisation of annual income of defaulted (0 and 1)")
# plt.show()

# plt.figure()
# plt.subplot(111)
# sns.boxplot(x = 'defaulted', y = 'annual_inc', data = loan_dataset_cp)
# plt.xlabel("Defaulted? (0 = 'No', 1 = 'Yes')")
# plt.ylabel("Annual Income")
# plt.title("Visualisation of annual income of defaulted (0 and 1)")
# plt.show()
```



REMOVE OUTLIERS

In [51]:
```
#let's find out the total population we have for the annual income and see if we
#take 99% out of it
```

In [52]:
```
total_annual_inc = loan_dataset_cp['annual_inc'].count()
```

In [53]:
```
top_99_total_annual_inc = round(total_annual_inc * 0.99)
```

In [54]:
```
top_99_total_annual_inc
```

Out[54]: 39320

In [55]:
```
#so we could create a dataframe out of these 39320 values
```

In [56]:
```
loan_dataset_cp_temp = pd.DataFrame({'defaulted':loan_dataset_cp.defaulted, 'ann
```

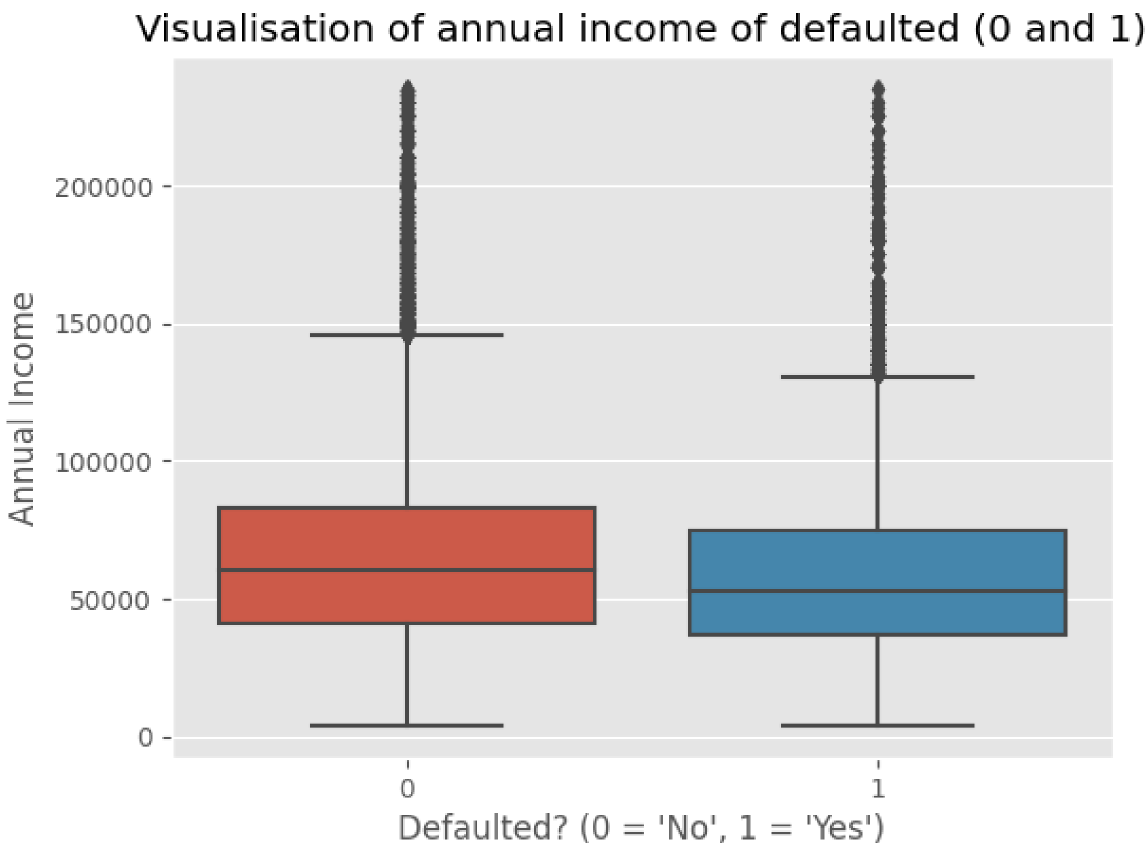
In [57]:
```
top_99 = loan_dataset_cp_temp.sort_values(by = 'annual_inc').head(top_99_total_a
top_99
```

Out[57]:

	defaulted	annual_inc
35501	0	4000.0
29283	1	4080.0
30726	0	4200.0
37709	0	4200.0
36639	0	4800.0
...	...	...
3475	0	234000.0
33036	0	234000.0
37048	0	234600.0
29878	0	234996.0
32316	1	235000.0

39320 rows × 2 columns

In [58]:
```
plt.figure()
plt.style.use('ggplot')
#we could also do: plt.style.use('seaborn')
sns.boxplot(x = 'defaulted', y = 'annual_inc', data = top_99)
plt.xlabel("Defaulted? (0 = 'No', 1 = 'Yes')")
plt.ylabel("Annual Income")
plt.title("Visualisation of annual income of defaulted (0 and 1)")
plt.show()
```



In [59]:
```
#let's try and take out top 5 % and repeat the whole visualization process again
```

In [60]:
```
top_95_total_annual_inc = round(total_annual_inc * 0.95)
```

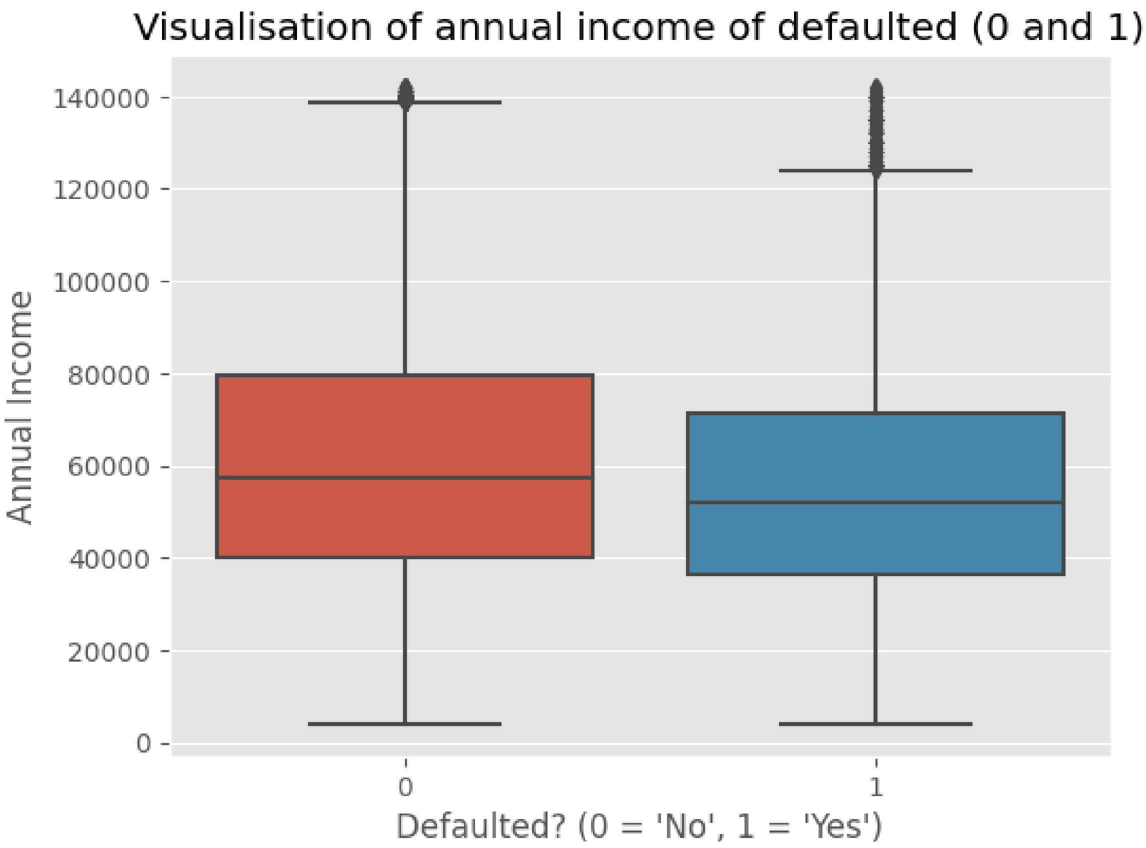
In [61]:
```
top_95 = loan_dataset_cp_temp.sort_values(by = 'annual_inc').head(top_95_total_a
top_95
```

Out[61]:

	defaulted	annual_inc
35501	0	4000.0
29283	1	4080.0
30726	0	4200.0
37709	0	4200.0
36639	0	4800.0
...	...	...
29171	0	141600.0
28537	1	141996.0
29447	0	141996.0
17651	0	141996.0
37220	0	142000.0

37731 rows × 2 columns

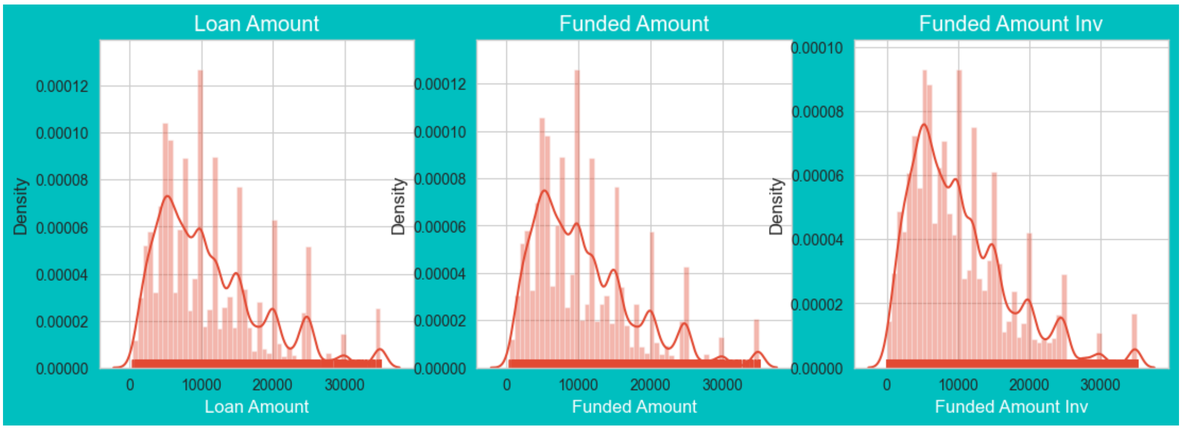
In [62]:
```
plt.figure()
plt.style.use('ggplot')
#we could also do: plt.style.use('seaborn')
sns.boxplot(x = 'defaulted', y = 'annual_inc', data = top_95)
plt.xlabel("Defaulted? (0 = 'No', 1 = 'Yes')")
plt.ylabel("Annual Income")
plt.title("Visualisation of annual income of defaulted (0 and 1)")
plt.show()

#No actual correlation between annual income to defaulters and non defaulters be
```



In [63]:
```
#Compare the distributions of three loan amounts fields the: loan_amnt, funded_a
```

In [64]:
```
fig = plt.figure(facecolor = 'c',
                figsize = (13,4)
                )
sns.set_style('whitegrid')

ax_1 = fig.add_subplot(131)
sns.distplot(loan_dataset_cp['loan_amnt'], ax = ax_1, rug = True)
ax_1.set_title("Loan Amount", color = 'white')
ax_1.set_xlabel("Loan Amount", fontsize = 12, color = 'white')

ax_2 = fig.add_subplot(132)
sns.distplot(loan_dataset_cp['funded_amnt'], ax = ax_2, rug = True)
ax_2.set_title("Funded Amount", color = 'white')
ax_2.set_xlabel("Funded Amount", fontsize = 12, color = 'white')

ax_3 = fig.add_subplot(133)
sns.distplot(loan_dataset_cp['funded_amnt_inv'], ax = ax_3, rug = True)
ax_3.set_title("Funded Amount Inv", color = 'white')
ax_3.set_xlabel("Funded Amount Inv", fontsize = 12, color = 'white')
```

```
# fig.tight_layout()
fig.show()

#I'm not really sure of what all these mean sha
```



In [65]:
```
#since interest rate is already, we know it's a percentage.., should we do the c
```

In [66]:
```
loan_dataset_cp.head()
```

Out[66]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	su
0	5000	5000	4975.0	36 months	10.65%	162.87	B	
1	2500	2500	2500.0	60 months	15.27%	59.83	C	
2	2400	2400	2400.0	36 months	15.96%	84.33	C	
3	10000	10000	10000.0	36 months	13.49%	339.31	C	
4	3000	3000	3000.0	60 months	12.69%	67.79	B	

In [67]:
```
type(loan_dataset_cp['int_rate'][0])
```

Out[67]: str

In [68]:
```
loan_dataset_cp['int_rate_converted'] = loan_dataset_cp['int_rate'].str.strip('%
```

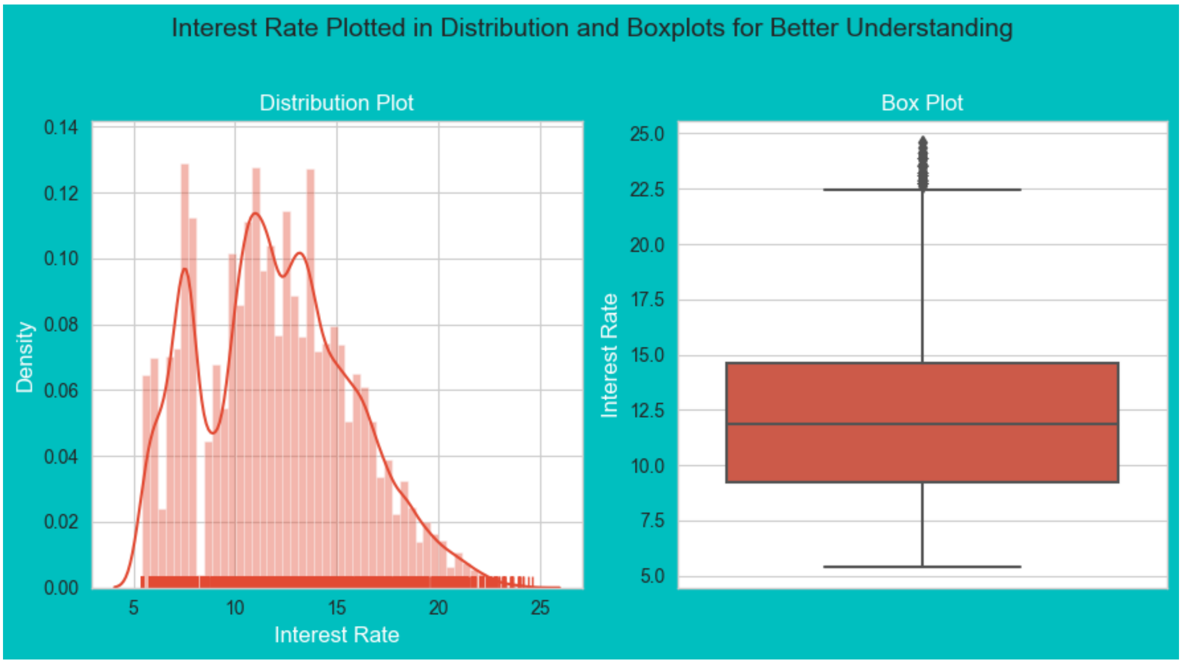
In [69]:
```
loan_dataset_cp.head()
```

Out[69]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	su
0	5000	5000	4975.0	36 months	10.65%	162.87	B	
1	2500	2500	2500.0	60 months	15.27%	59.83	C	
2	2400	2400	2400.0	36 months	15.96%	84.33	C	
3	10000	10000	10000.0	36 months	13.49%	339.31	C	
4	3000	3000	3000.0	60 months	12.69%	67.79	B	

In [70]:

```
fig = plt.figure(figsize=(9,5), facecolor = 'c')
sns.set_style('whitegrid')

#adding subplots
#distribution plots
ax_1 = fig.add_subplot(121)
sns.distplot(loan_dataset_cp['int_rate_converted'], ax = ax_1, rug = True)
ax_1.set_title('Distribution Plot', fontsize = 12, color = 'w')
ax_1.set_xlabel('Interest Rate', color = 'w')
ax_1.set_ylabel('Density', color = 'w')

#box plots
ax_2 = fig.add_subplot(122)
sns.boxplot(loan_dataset_cp['int_rate_converted'], ax = ax_2)
ax_2.set_title('Box Plot', fontsize = 12, color = 'w')
ax_2.get_xaxis().set_visible(False)
ax_2.set_ylabel('Interest Rate', color = 'w')

fig.suptitle('Interest Rate Plotted in Distribution and Boxplots for Better Unde
fig.tight_layout(rect = [0,0,1,0.96])

fig.show()

#Observations:

#Using the boxplot as a reference  most of the interest rate are concentrated be
```



In [71]:
```
#let's continue our visualisation
#we want to see the count of people that defaulted and those that didn't
#this time around we are using the loan status column and not the defaulted colu
#to grasp a better understanding of what we are doing
```

In [72]:
```
s = loan_dataset_cp['loan_status'].value_counts()
```

In [73]:
```
tot = s.sum()
```

In [74]:
```
#wow, come and see a magic
#reset index transforms a series to a data frame.., just take a look at both of
```

In [75]:
```
s
```

Out[75]:
```
loan_status
Fully Paid      32950
Charged Off      5627
Current          1140
Name: count, dtype: int64
```

In [76]:
```
s.reset_index()
```

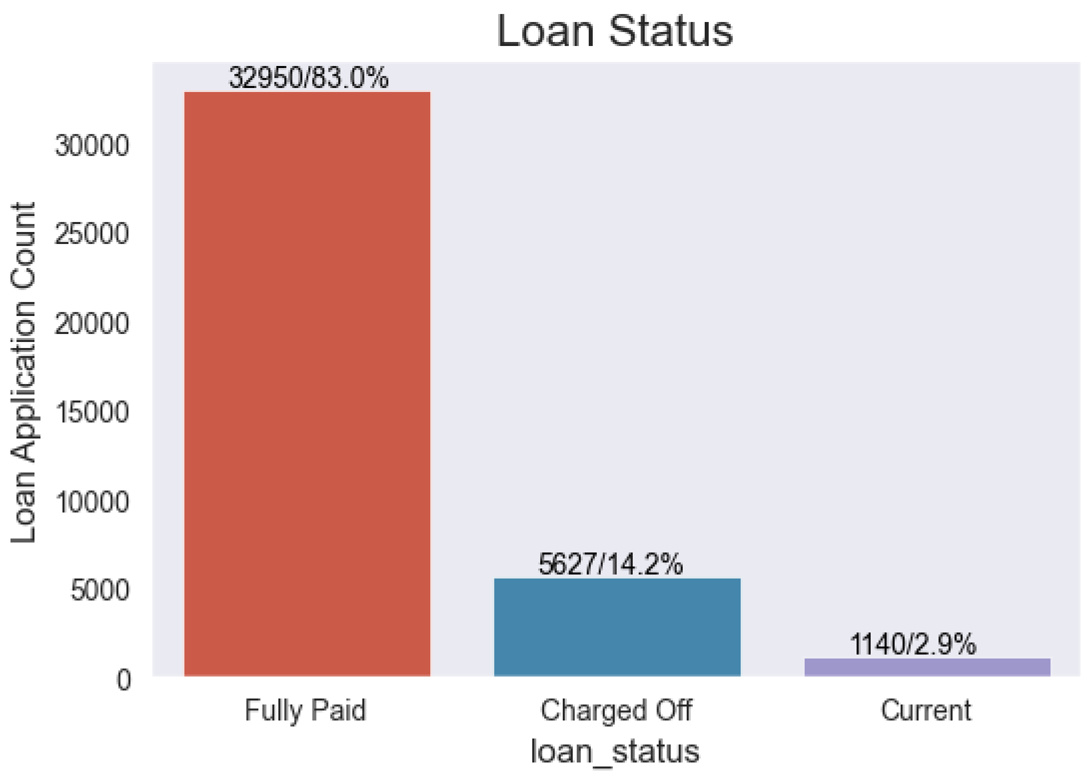
Out[76]:

	loan_status	count
0	Fully Paid	32950
1	Charged Off	5627
2	Current	1140

In [77]:
```
for i,v in s.reset_index().iterrows():
    text = str(v['count']) + "/" + str(round((v['count']/tot)*100, 1)) + '%'
    print(text)
```

```
32950/83.0%
5627/14.2%
1140/2.9%
```

In [78]:
```
text
```
Out[78]: '1140/2.9%'

In [79]:
```
#Let's do a count plot of all the loan status alright
```
In [80]:
```
plt.figure(figsize = (6,4))
sns.set_style('dark')
sns.countplot(data = loan_dataset_cp, x = "loan_status")
plt.ylabel("Loan Application Count")
plt.title('Loan Status', fontsize = 16)

s = loan_dataset_cp['loan_status'].value_counts()
tot = s.sum()

for i,v in s.reset_index().iterrows():
    text = str(v['count']) + "/" + str(round((v['count']/tot)*100, 1)) + '%'
    plt.text(i-0.25, v['count'] + 200, text, color = 'k')

plt.show()
```


MULTIVARIATE ANALYSIS

In [81]:
```
#FIRST OFF - Let's see the correlation between the different columns in the data
```
In [82]:
```
corr_column_enabled = [column for column in loan_dataset_cp.columns if loan_data

# for column in loan_dataset_cp.columns:
#     if loan_dataset_cp[column].dtype == 'int64' or loan_dataset_cp[column].dty
#         print(column)
```
In [83]:
```
corr_column_enabled
```

Out[83]:
```
['loan_amnt',
 'funded_amnt',
 'funded_amnt_inv',
 'installment',
 'annual_inc',
 'dti',
 'inq_last_6mths',
 'open_acc',
 'pub_rec',
 'revol_bal',
 'total_acc',
 'pub_rec_bankruptcies',
 'defaulted',
 'int_rate_converted']
```

In [84]:
```
loan_dataset_cp_corr_enabled = loan_dataset_cp[['loan_amnt',
 'funded_amnt',
 'funded_amnt_inv',
 'installment',
 'annual_inc',
 'dti',
 'inq_last_6mths',
 'open_acc',
 'pub_rec',
 'revol_bal',
 'total_acc',
 'pub_rec_bankruptcies',
 'defaulted',
 'int_rate_converted']]
```

In [85]:
```
#we could also do this to retrieve the columns with int and float 64 dtype
# loan_dataset_cp.columns[loan_dataset_cp.dtypes == 'object']
#loan_dataset_cp[f[(f == 'int64') | (f== 'float64')].index]
#where f = loan_dataset_cp.dtypes
```

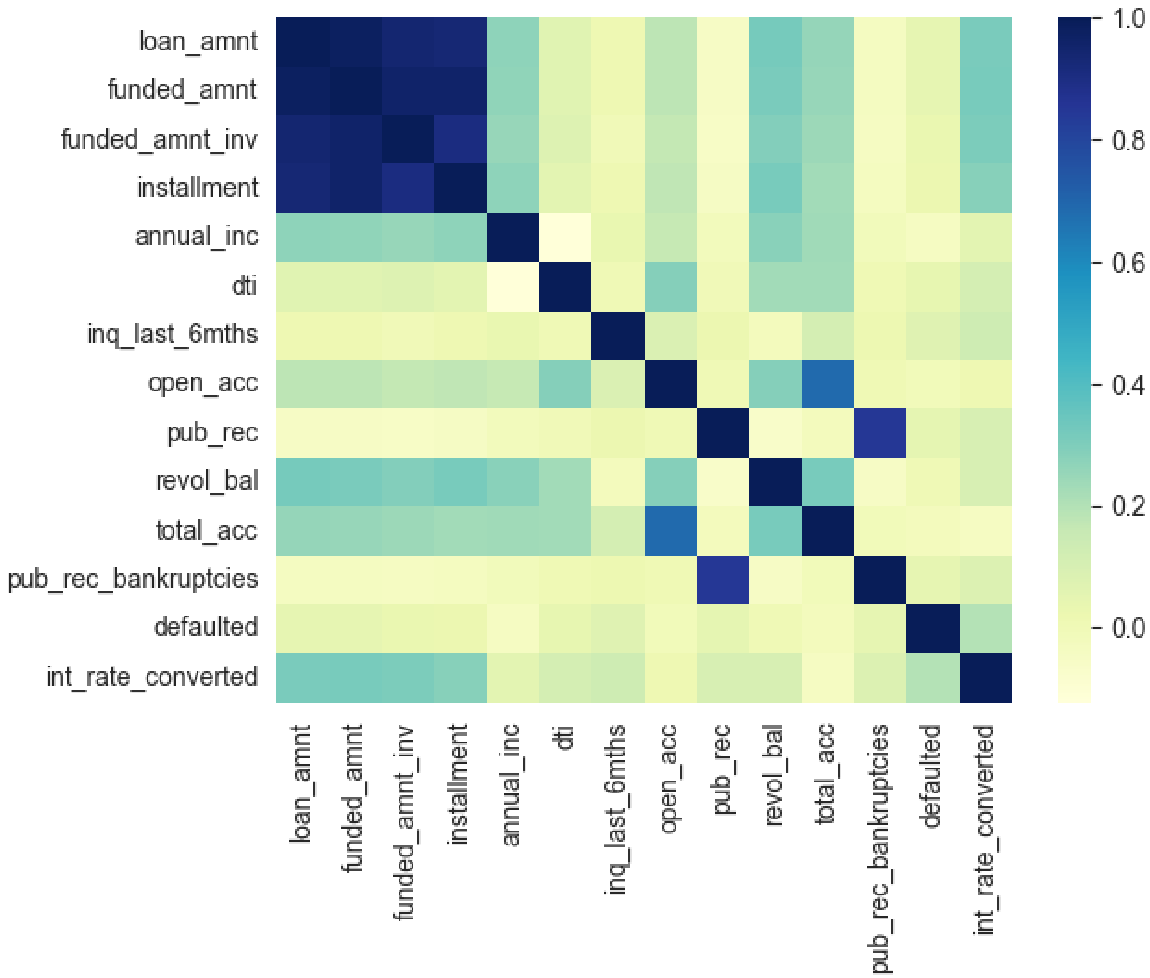
In [86]:
```
loan_dataset_cp_corr_enabled.corr()
```

Out[86]:

	loan_amnt	funded_amnt	funded_amnt_inv	installment	annual_ir
loan_amnt	1.000000	0.981578	0.940034	0.930288	0.27114
funded_amnt	0.981578	1.000000	0.958422	0.956159	0.26696
funded_amnt_inv	0.940034	0.958422	1.000000	0.905039	0.25437
installment	0.930288	0.956159	0.905039	1.000000	0.27087
annual_inc	0.271149	0.266965	0.254375	0.270874	1.00000
dti	0.066439	0.066283	0.074689	0.054186	-0.12273
inq_last_6mths	0.009229	0.009259	-0.005712	0.009722	0.03390
open_acc	0.177168	0.175530	0.163027	0.172812	0.15820
pub_rec	-0.051236	-0.052169	-0.053214	-0.046532	-0.01868
revol_bal	0.317597	0.310392	0.290797	0.312679	0.27996
total_acc	0.256442	0.250589	0.242854	0.230824	0.23577
pub_rec_bankruptcies	-0.037180	-0.038502	-0.042746	-0.034103	-0.01680
defaulted	0.048217	0.045544	0.026621	0.022589	-0.04166
int_rate_converted	0.309415	0.312619	0.306657	0.282703	0.05318

In [87]:

```
sns.heatmap(loan_dataset_cp_corr_enabled.corr(), cmap = 'YlGnBu')
#other cmaps available are : cmap = 'viridis', cmap = 'plasma', cmap = 'cubeheli
plt.show()
#So from the heatmap, we could see that there is a strong correlation between th
#funded_amnt_inv, and investment

#there is also a strong relationship between public_rec and public_rec bankruptc
```



In [88]:
```
#LET'S FIGURE OUT THE COUNT OF NULL VALUES IN OUR DATAFRAME
```

In [89]:
```
loan_dataset_cp.isnull().sum()
#we could have also done loan_dataset_cp.isna().sum()
```

Out[89]:
```
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
annual_inc                   0
verification_status          0
issue_d                      0
loan_status                  0
purpose                      0
zip_code                     0
addr_state                   0
dti                          0
inq_last_6mths               0
open_acc                     0
pub_rec                      0
revol_bal                    0
revol_util                  50
total_acc                    0
last_pymnt_d                71
pub_rec_bankruptcies       697
defaulted                    0
int_rate_converted           0
dtype: int64
```

In [90]:
```
a = (loan_dataset_cp.isnull().sum()/loan_dataset_cp.shape[0]) * 100
#the percentage of null values in the dataframe
```

In [91]:
```
b = pd.DataFrame(a[a>0.05], columns = ['Percentage_null_values'])
```

In [92]:
```
b.sort_values(by = 'Percentage_null_values', ascending = False)
```

Out[92]:

	Percentage_null_values
emp_title	6.191303
emp_length	2.706650
pub_rec_bankruptcies	1.754916
last_pymnt_d	0.178765
revol_util	0.125891

In [93]:
```
loan_dataset_cp.columns[loan_dataset_cp.dtypes == 'object'] #this is a life save
```

Out[93]:
```
Index(['term', 'int_rate', 'grade', 'sub_grade', 'emp_title', 'emp_length',
       'verification_status', 'issue_d', 'loan_status', 'purpose', 'zip_code',
       'addr_state', 'revol_util', 'last_pymnt_d'],
      dtype='object')
```

In [94]:
```
loan_dataset_cp.term.value_counts()
```

Out[94]:
```
term
 36 months    29096
 60 months    10621
Name: count, dtype: int64
```

Unique values in each categorical feature

In [95]:
```
#let's get a count of the unique not numeric features in our data set
```

In [96]:
```
for i in loan_dataset_cp.columns[loan_dataset_cp.dtypes == 'object']:
    print(loan_dataset_cp[i].value_counts())
    print('--------------------------------\n-----------------------------------
```

```
term
 36 months    29096
 60 months    10621
Name: count, dtype: int64
--------------------------------
-----------------------------------
int_rate
10.99%    956
13.49%    826
11.49%    825
7.51%     787
7.88%     725
         ...
18.36%      1
16.96%      1
16.15%      1
16.01%      1
17.44%      1
Name: count, Length: 371, dtype: int64
--------------------------------
-----------------------------------
grade
B    12020
A    10085
C     8098
D     5307
E     2842
F     1049
G      316
Name: count, dtype: int64
--------------------------------
-----------------------------------
sub_grade
B3    2917
A4    2886
A5    2742
B5    2704
B4    2512
C1    2136
B2    2057
C2    2011
B1    1830
A3    1810
C3    1529
A2    1508
D2    1348
C4    1236
C5    1186
D3    1173
A1    1139
D4     981
D1     931
D5     874
E1     763
E2     656
E3     553
E4     454
E5     416
F1     329
F2     249
```

```
F3     185
F4     168
F5     118
G1     104
G2      78
G4      56
G3      48
G5      30
Name: count, dtype: int64
--------------------------------
-----------------------------------
emp_title
US Army                             134
Bank of America                     109
IBM                                  66
AT&T                                 59
Kaiser Permanente                    56
                                   ...
Community College of Philadelphia     1
AMEC                                  1
lee county sheriff                    1
Bacon County Board of Education       1
Evergreen Center                      1
Name: count, Length: 28820, dtype: int64
--------------------------------
-----------------------------------
emp_length
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: count, dtype: int64
--------------------------------
-----------------------------------
verification_status
Not Verified       16921
Verified           12809
Source Verified     9987
Name: count, dtype: int64
--------------------------------
-----------------------------------
issue_d
Dec-11    2260
Nov-11    2223
Oct-11    2114
Sep-11    2063
Aug-11    1928
Jul-11    1870
Jun-11    1827
May-11    1689
Apr-11    1562
Mar-11    1443
Jan-11    1380
```

```
Feb-11    1297
Dec-10    1267
Oct-10    1132
Nov-10    1121
Jul-10    1119
Sep-10    1086
Aug-10    1078
Jun-10    1029
May-10     920
Apr-10     827
Mar-10     737
Feb-10     627
Nov-09     602
Dec-09     598
Jan-10     589
Oct-09     545
Sep-09     449
Aug-09     408
Jul-09     374
Jun-09     356
May-09     319
Apr-09     290
Mar-09     276
Feb-09     260
Jan-09     239
Mar-08     236
Dec-08     223
Nov-08     184
Feb-08     174
Jan-08     171
Apr-08     155
Oct-08      96
Dec-07      85
Jul-08      83
May-08      71
Aug-08      71
Jun-08      66
Oct-07      47
Nov-07      37
Aug-07      33
Sep-08      32
Jul-07      30
Sep-07      18
Jun-07       1
Name: count, dtype: int64
--------------------------------
-----------------------------------
loan_status
Fully Paid     32950
Charged Off     5627
Current         1140
Name: count, dtype: int64
--------------------------------
-----------------------------------
purpose
debt_consolidation    18641
credit_card            5130
other                  3993
home_improvement       2976
major_purchase         2187
```

```
small_business         1828
car                    1549
wedding                 947
medical                 693
moving                  583
vacation                381
house                   381
educational             325
renewable_energy        103
Name: count, dtype: int64
--------------------------------
-----------------------------------
zip_code
100xx    597
945xx    545
112xx    516
606xx    503
070xx    473
        ...
381xx      1
378xx      1
739xx      1
396xx      1
469xx      1
Name: count, Length: 823, dtype: int64
--------------------------------
-----------------------------------
addr_state
CA    7099
NY    3812
FL    2866
TX    2727
NJ    1850
IL    1525
PA    1517
VA    1407
GA    1398
MA    1340
OH    1223
MD    1049
AZ     879
WA     840
CO     792
NC     788
CT     751
MI     720
MO     686
MN     615
NV     497
SC     472
WI     460
AL     452
OR     451
LA     436
KY     325
OK     299
KS     271
UT     258
AR     245
DC     214
```

```
RI     198
NM     189
WV     177
HI     174
NH     171
DE     114
MT      85
WY      83
AK      80
SD      64
VT      54
MS      19
TN      17
IN       9
ID       6
IA       5
NE       5
ME       3
Name: count, dtype: int64
--------------------------------
-----------------------------------
revol_util
0%        977
0.20%      63
63%        62
40.70%     58
66.70%     58
         ...
25.74%      1
47.36%      1
24.65%      1
10.61%      1
7.28%       1
Name: count, Length: 1089, dtype: int64
--------------------------------
-----------------------------------
last_pymnt_d
May-16    1256
Mar-13    1026
Dec-14     945
May-13     907
Feb-13     869
          ...
Jun-08      10
Nov-08      10
Mar-08       5
Jan-08       4
Feb-08       1
Name: count, Length: 101, dtype: int64
--------------------------------
-----------------------------------
```

In [97]:
```
len(loan_dataset_cp.annual_inc.values)
```

Out[97]:
```
39717
```

In [98]:
```
bins = [0, 30000, 60000, 100000, 500000]
```

In [99]:
```
group_names = ['0-30K', '30-60K', '60-100K', '100K+']
```

In [100…
```
cat = pd.cut(loan_dataset_cp.annual_inc.values, bins, labels = group_names)
```

In [101…
```
cat
```

Out[101…
```
['0-30K', '0-30K', '0-30K', '30-60K', '60-100K', ..., '100K+', '0-30K', '60-100
K', '100K+', '0-30K']
Length: 39717
Categories (4, object): ['0-30K' < '30-60K' < '60-100K' < '100K+']
```

In [102…
```
pd.value_counts(cat)
```

Out[102…
```
30-60K     16861
60-100K    12545
100K+       5620
0-30K       4624
Name: count, dtype: int64
```

In [103…
```
#now this is very good, but what we actually want is a tag to the dataset.., mor
```

In [104…
```
# loan_dataset_cp.annual_inc.quantile(q=1)

# quantile it might be beneficial
```

In [105…
```
def classify_annual_inc(x):
    if x >0 and x <= 30000:
        return '0-30K'
    elif x > 30000 and x <= 60000:
        return '30-60K'
    elif x > 60000 and x <= 100000:
        return '60-100K'
    else:
        return '100K+'
```

In [106…
```
loan_dataset_cp.annual_inc.dtype
```

Out[106…
```
dtype('float64')
```

In [107…
```
loan_dataset_cp['binned_annual_inc'] = loan_dataset_cp['annual_inc'].astype(int)
```

In [108…
```
loan_dataset_cp.head()
```

Out[108…

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	su
0	5000	5000	4975.0	36 months	10.65%	162.87	B	
1	2500	2500	2500.0	60 months	15.27%	59.83	C	
2	2400	2400	2400.0	36 months	15.96%	84.33	C	
3	10000	10000	10000.0	36 months	13.49%	339.31	C	
4	3000	3000	3000.0	60 months	12.69%	67.79	B	

In [109…

```
#let's use cross tab to get the number of people that falls within the annual_in
```

In [110…

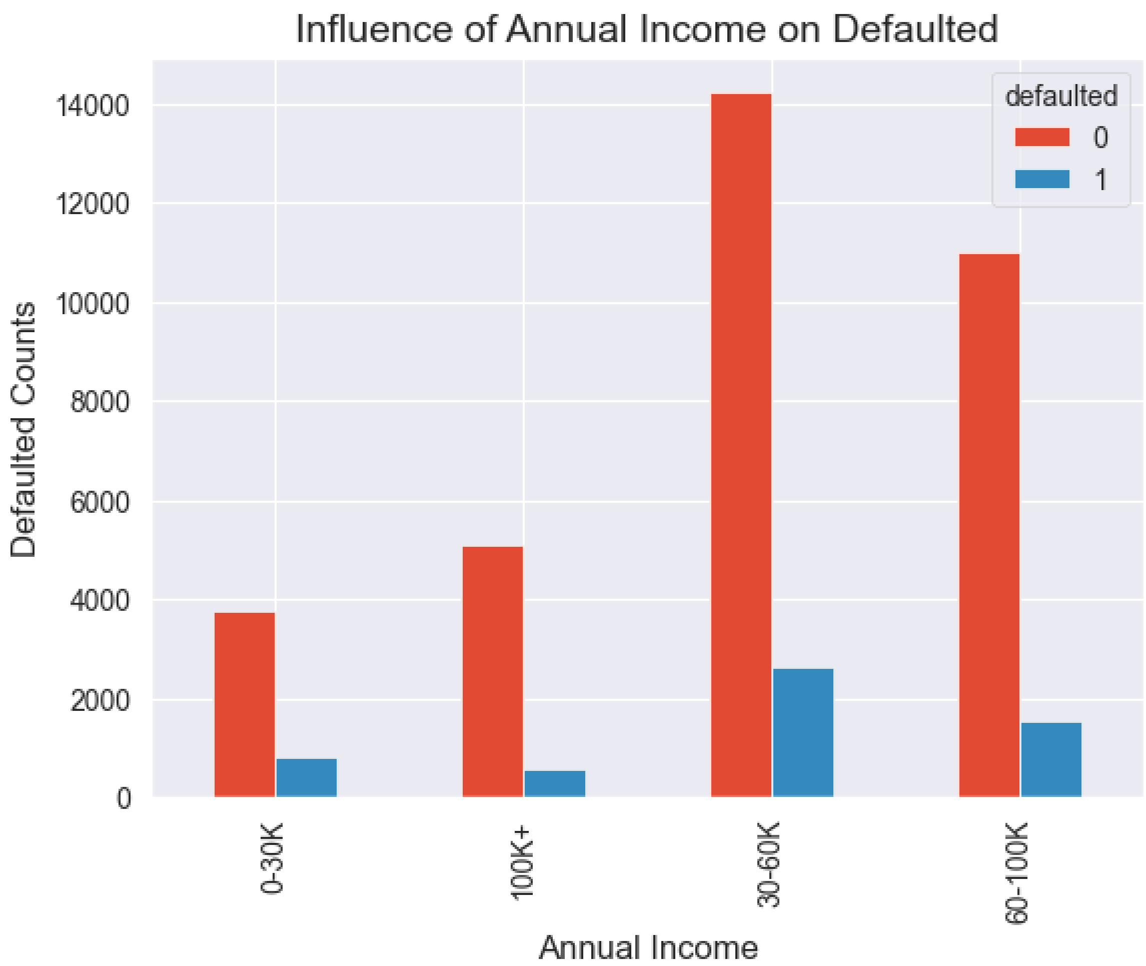
```
pd.crosstab(loan_dataset_cp['binned_annual_inc'], loan_dataset_cp['defaulted'])
```

Out[110…

defaulted	0	1
binned_annual_inc		
0-30K	3785	839
100K+	5095	592
30-60K	14220	2641
60-100K	10990	1555

In [111…

```
#let's plot it
pd.crosstab(loan_dataset_cp['binned_annual_inc'], loan_dataset_cp['defaulted']).
plt.title("Influence of Annual Income on Defaulted", fontsize = 14)
plt.xlabel("Annual Income")
plt.ylabel("Defaulted Counts")
plt.grid()
plt.show()
```



In [112…
```
df_annual_inc_defaulters = loan_dataset_cp[loan_dataset_cp.defaulted == 1].group
```

In [113…
```
df_annual_inc_defaulted = loan_dataset_cp.groupby('binned_annual_inc')['defaulte
```

In [114…
```
df_annual_inc_full = pd.merge(df_annual_inc_defaulters, df_annual_inc_defaulted,
         #left_on = 'binned_annual_inc', right_on = 'binned_annual_inc',
         on = 'binned_annual_inc',
         how = 'left',
         suffixes = ('_yes', '_yes_and_no'))
```

In [115…
```
df_annual_inc_full['Ratio'] = (df_annual_inc_full.defaulted_yes/df_annual_inc_fu
```

In [116…
```
df_annual_inc_full.Ratio = df_annual_inc_full.Ratio.transform(lambda x: x*100)
```

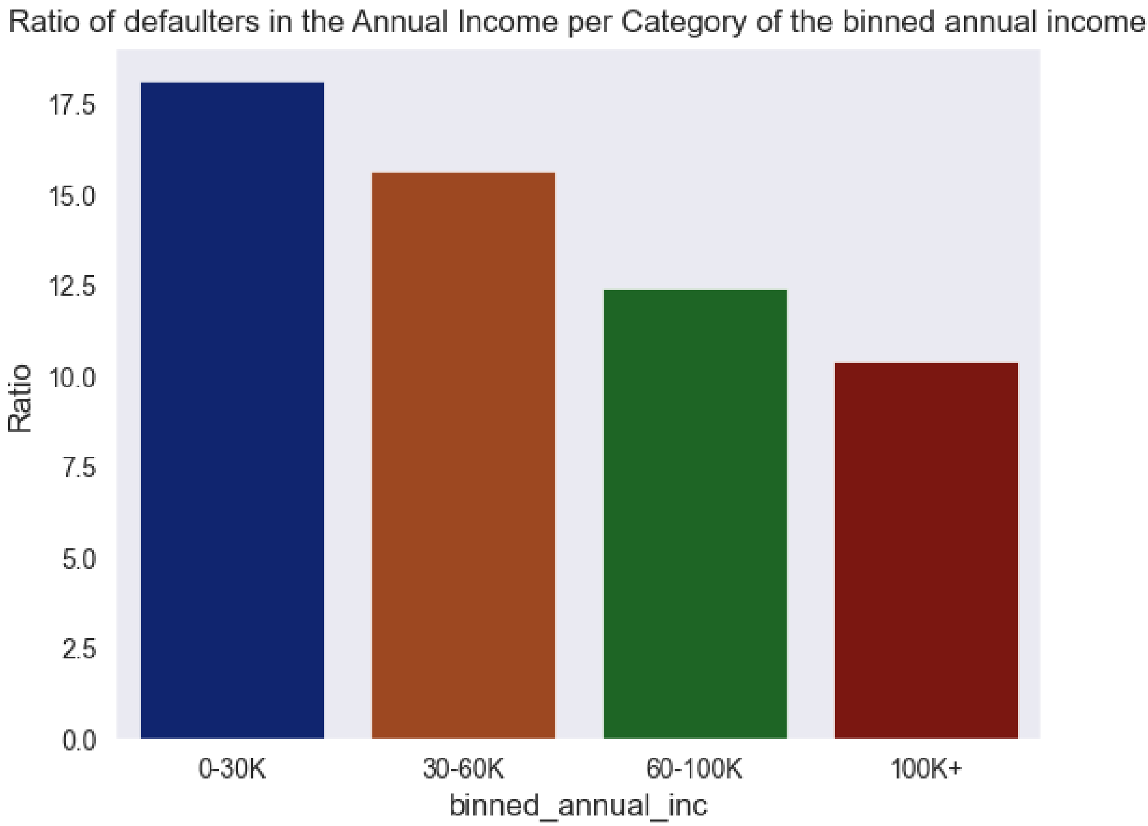
In [117…
```
df_annual_inc_full
```

Out[117…

	binned_annual_inc	defaulted_yes	defaulted_yes_and_no	Ratio
0	0-30K	839	4624	18.144464
1	100K+	592	5687	10.409706
2	30-60K	2641	16861	15.663365
3	60-100K	1555	12545	12.395377

In [118…
```
#let's see a  barplot of the binned_annual_inc and the ratio
```

```
plt.figure()
sns.set_style('dark')
sns.barplot(data = df_annual_inc_full.sort_values(by = 'Ratio', ascending = Fals
plt.title("Ratio of defaulters in the Annual Income per Category of the binned a
plt.show()
```



In [119…

```
#alright.., let's apply some binning on the funding amount.
#let's start by taking a look at the funded amount column's description
```

In [120…

```
def classify_funded_amount(x):
    if x > 0 and x < 10000:
        return '0-10K'
    elif x >= 10000 and x < 20000:
        return '10-20K'
    else:
        return '20K+'
```

In [121…

```
loan_dataset_cp['funded_amount_classification'] = loan_dataset_cp['funded_amnt']
```

In [122…

```
loan_dataset_cp.head()
```

Out[122...

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	su
0	5000	5000	4975.0	36 months	10.65%	162.87	B	
1	2500	2500	2500.0	60 months	15.27%	59.83	C	
2	2400	2400	2400.0	36 months	15.96%	84.33	C	
3	10000	10000	10000.0	36 months	13.49%	339.31	C	
4	3000	3000	3000.0	60 months	12.69%	67.79	B	

In [123...

```
#alright..., now let's find out the ratio of people that defaulted based on the
```

In [124...

```
df_funded_amnt_defaulters = loan_dataset_cp[loan_dataset_cp.defaulted == 1].grou
```

In [125...

```
df_funded_amnt_defaulted = loan_dataset_cp.groupby('funded_amount_classification
```

In [126...

```
df_funded_amnt_defaulted
```

Out[126...

	funded_amount_classification	defaulted
0	0-10K	20071
1	10-20K	14060
2	20K+	5586

In [127...

```
df_funded_amnt_full = pd.merge(df_funded_amnt_defaulters, df_funded_amnt_default
         on = 'funded_amount_classification',
         how = 'left',
         suffixes = ('_yes', '_yes_and_no'))
```

In [128...

```
df_funded_amnt_full
```

Out[128...

	funded_amount_classification	defaulted_yes	defaulted_yes_and_no
0	0-10K	2647	20071
1	10-20K	2013	14060
2	20K+	967	5586

In [129...

```
df_funded_amnt_full['Ratio'] = (df_funded_amnt_full.defaulted_yes/df_funded_amnt
```

In [130...

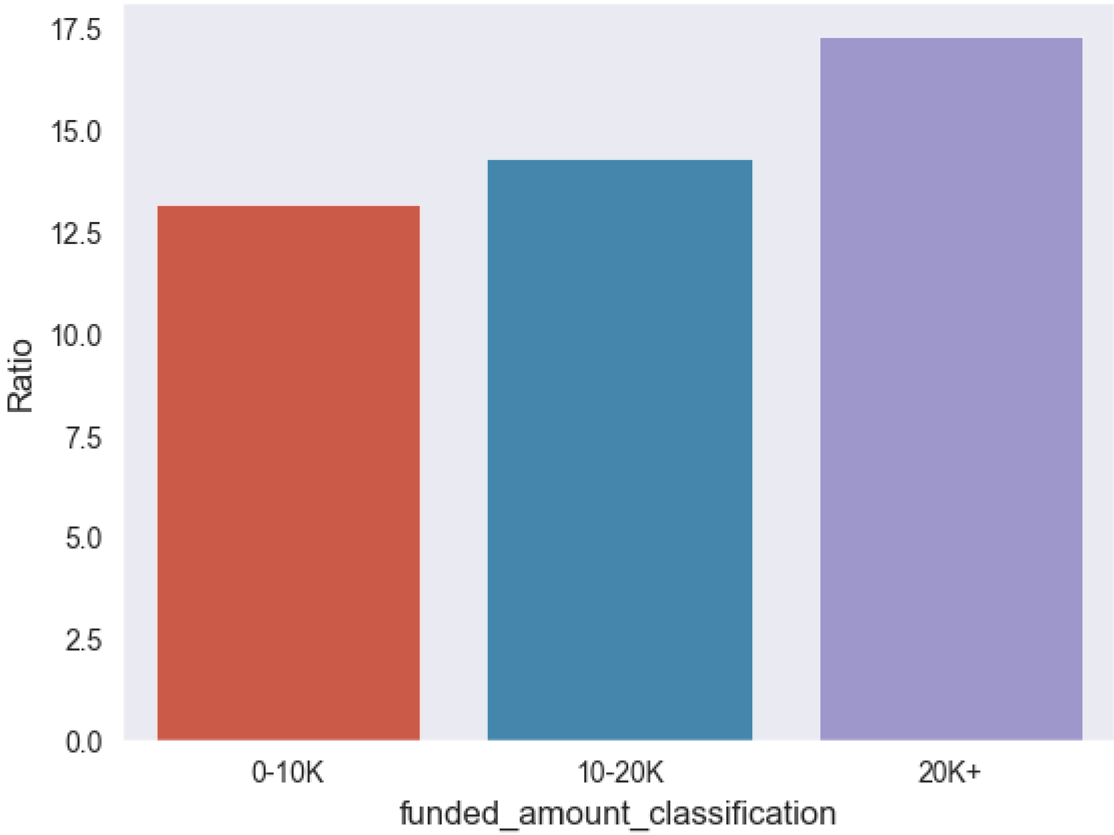
```
df_funded_amnt_full
```

Out[130…

	funded_amount_classification	defaulted_yes	defaulted_yes_and_no	Ratio
0	0-10K	2647	20071	13.188182
1	10-20K	2013	14060	14.317212
2	20K+	967	5586	17.311135

In [131…

```
#let's plot and see the visualisation
sns.barplot(data = df_funded_amnt_full, x = 'funded_amount_classification', y =
plt.show()
```



In [132…

```
#let's get a kind of a more complex visualisation
#in each category of the funded amount, let's get the annual income those folks
#won't that be nicer or should i say better
```

In [133…

```
fmnt_ainc_defaulters = loan_dataset_cp[loan_dataset_cp.defaulted == 1].groupby([
```

In [134…

```
fmnt_ainc_defaulted = loan_dataset_cp.groupby(['funded_amount_classification', '
```

In [135…

```
fmnt_ainc_full = pd.merge(fmnt_ainc_defaulters, fmnt_ainc_defaulted, on = ['fund
```

In [136…

```
fmnt_ainc_full = fmnt_ainc_full.assign(Ratio = fmnt_ainc_full.defaulted_yes/fmnt
```

In [137…

```
fmnt_ainc_full
```

Out[137…

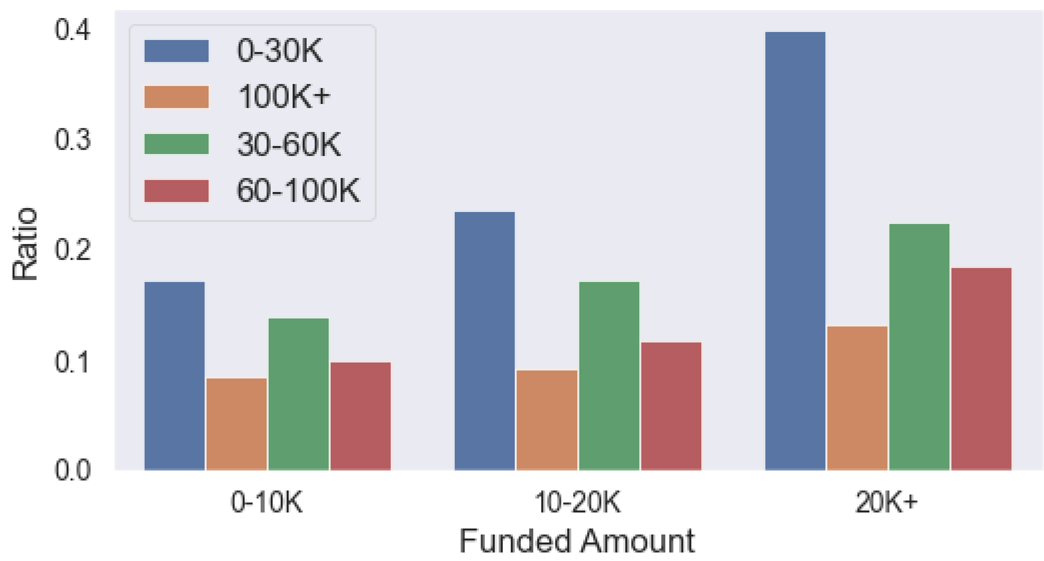
	funded_amount_classification	binned_annual_inc	defaulted_yes	defaulted_yes_and_n
0	0-10K	0-30K	693	400
1	0-10K	100K+	127	151
2	0-10K	30-60K	1338	961
3	0-10K	60-100K	489	493
4	10-20K	0-30K	144	61
5	10-20K	100K+	197	213
6	10-20K	30-60K	1065	618
7	10-20K	60-100K	607	512
8	20K+	0-30K	2	
9	20K+	100K+	268	204
10	20K+	30-60K	238	105
11	20K+	60-100K	459	248

In [138…

```
#this is great stuff.., now let's proceed and see what we have alright...., let'
```

In [139…

```
plt.figure(figsize=(6,3))
sns.barplot(data = fmnt_ainc_full, x = 'funded_amount_classification', y = 'Rati
            hue = 'binned_annual_inc', palette= 'deep'
           )
plt.xlabel('Funded Amount', fontsize = 12)
plt.ylabel('Ratio', fontsize = 12)
plt.legend(fontsize = 12)
plt.rc('xtick', labelsize = 10)

#look more of plt.xticks and plt.yticks..., mine is kind of like a number
plt.show()
```



In [140…

```
loan_dataset_cp[loan_dataset_cp.columns[(loan_dataset_cp.dtypes == 'int64')|(loa
```

Out[140…

	loan_amnt	funded_amnt	funded_amnt_inv	installment	annual_inc	dti	in
defaulted							
0	11073.0	10815.0	10320.0	323.0	70049.0	13.0	
1	12104.0	11753.0	10865.0	336.0	62427.0	14.0	

In [141…

```
loan_dataset_cp
```

Out[141…

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade
0	5000	5000	4975.0	36 months	10.65%	162.87	E
1	2500	2500	2500.0	60 months	15.27%	59.83	C
2	2400	2400	2400.0	36 months	15.96%	84.33	C
3	10000	10000	10000.0	36 months	13.49%	339.31	C
4	3000	3000	3000.0	60 months	12.69%	67.79	E
...	...	...	...	...	...	...	..
39712	2500	2500	1075.0	36 months	8.07%	78.42	A
39713	8500	8500	875.0	36 months	10.28%	275.38	C
39714	5000	5000	1325.0	36 months	8.07%	156.84	A
39715	5000	5000	650.0	36 months	7.43%	155.38	A
39716	7500	7500	800.0	36 months	13.75%	255.43	E

39717 rows × 30 columns

In [142…

```
#is there any significant difference between the requested loan and what was dis
#calculated in percentage
print(((loan_dataset_cp.loan_amnt - loan_dataset_cp.funded_amnt)/(loan_dataset_c
print(((loan_dataset_cp.loan_amnt - loan_dataset_cp.funded_amnt)/(loan_dataset_c

#and for some cases what what was disbursed was little compared to what was requ
```

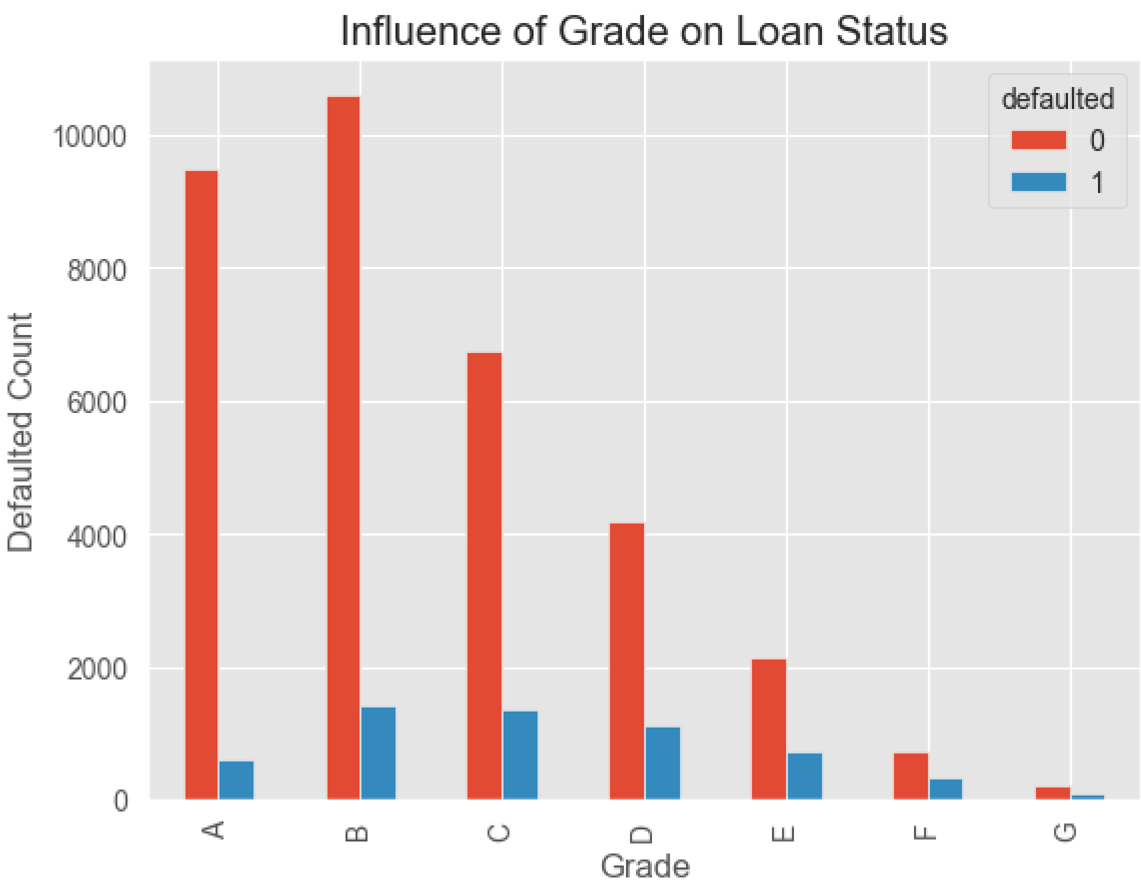
```
23556    89.875000
23296    89.750000
23288    88.928571
23416    86.750000
23337    84.750000
dtype: float64
13490    0.0
13491    0.0
13492    0.0
13493    0.0
39716    0.0
dtype: float64
```

CORRELATION BETWEEN THE GRADE OF LOANS AND DEFAULTED

In [143…

```
plt.figure()
plt.style.use('ggplot')
ct = pd.crosstab(loan_dataset_cp['grade'], loan_dataset_cp['defaulted'])
ct.plot(kind = 'bar')
plt.title('Influence of Grade on Loan Status')
plt.xlabel('Grade')
plt.ylabel('Defaulted Count')
plt.rc('xtick', labelsize = 10)
plt.rc('ytick', labelsize = 10)
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



In [144…

```
ct = ct.assign(ratio = (ct[1]/ct[0])*100)
```

In [145…

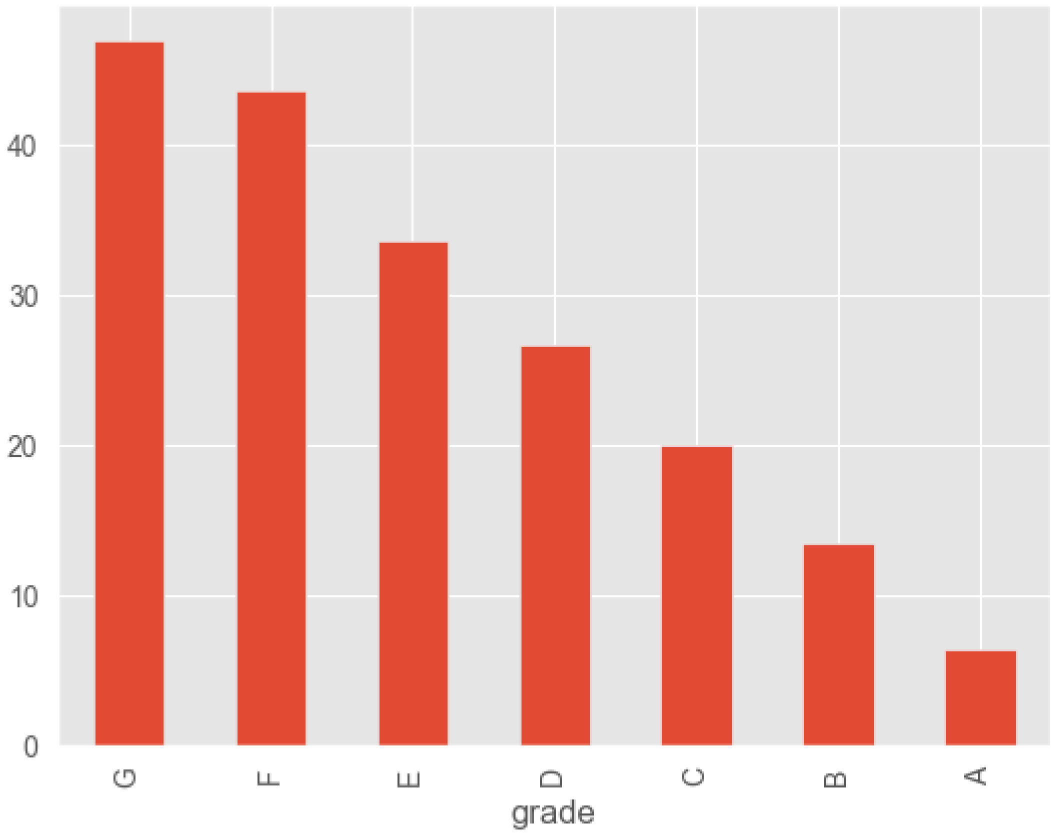
```
ct
```

Out[145…

defaulted	0	1	ratio
grade			
A	9483	602	6.348202
B	10595	1425	13.449740
C	6751	1347	19.952600
D	4189	1118	26.688947
E	2127	715	33.615421
F	730	319	43.698630
G	215	101	46.976744

In [146…

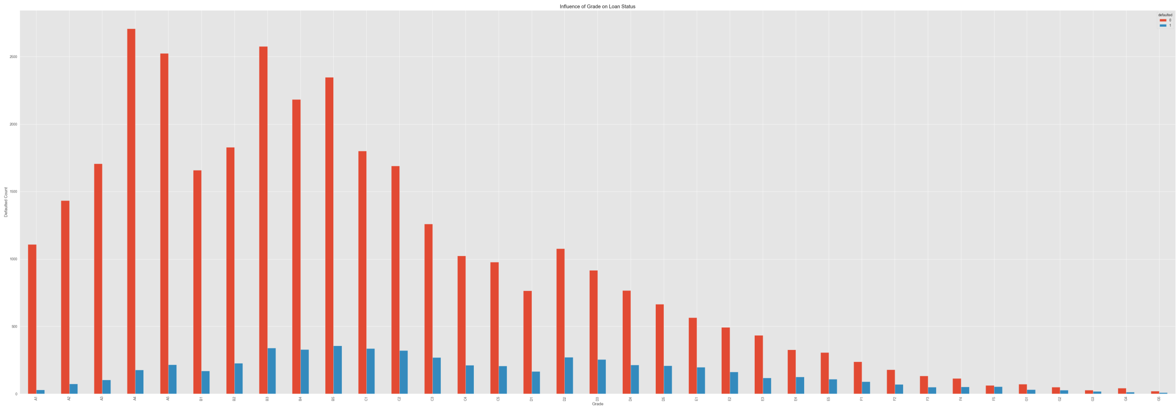
```
ct.ratio.sort_values(ascending = False).plot(kind = 'bar')
plt.show()
#from this visualisation it is evident that people defaulted more for a loan typ
```



In [147…

```
#what about the subgrades.., let's take a look at those..
plt.figure()
plt.style.use('ggplot')
ct_sub = pd.crosstab(loan_dataset_cp['sub_grade'], loan_dataset_cp['defaulted'])
ct_sub.plot(kind = 'bar', figsize = (60,20))
plt.title('Influence of Grade on Loan Status')
plt.xlabel('Grade')
plt.ylabel('Defaulted Count')
plt.rc('xtick', labelsize = 10)
plt.rc('ytick', labelsize = 10)
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



In [148…
```
#we can take a look at them and understand what's actually the ratio
```

In [149…
```
ct_sub['ratio'] = (ct_sub[1]/ct_sub[0] * 100).round(2)
```

In [150…
```
ct_sub
```

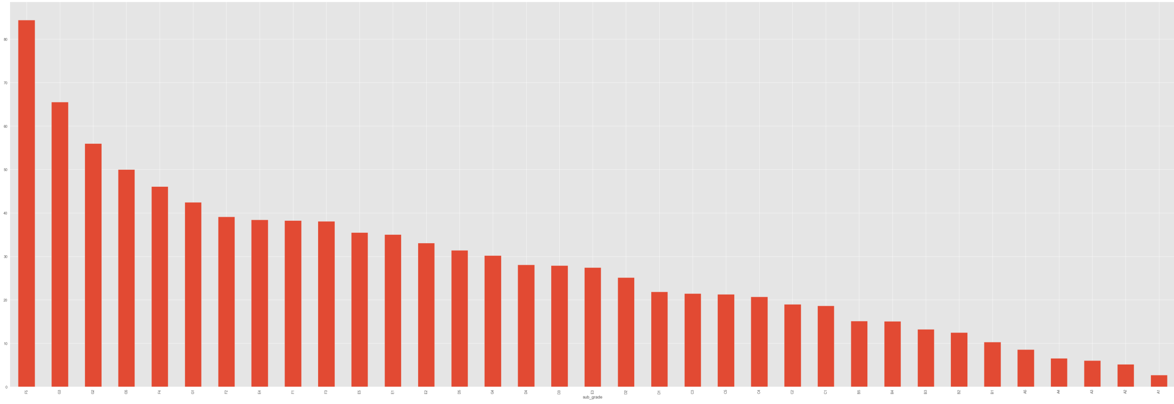
Out[150…

defaulted	0	1	ratio
sub_grade			
A1	1109	30	2.71
A2	1434	74	5.16
A3	1707	103	6.03
A4	2708	178	6.57
A5	2525	217	8.59
B1	1659	171	10.31
B2	1829	228	12.47
B3	2576	341	13.24
B4	2183	329	15.07
B5	2348	356	15.16
C1	1800	336	18.67
C2	1690	321	18.99
C3	1259	270	21.45
C4	1024	212	20.70
C5	978	208	21.27
D1	764	167	21.86
D2	1077	271	25.16
D3	917	256	27.92
D4	766	215	28.07
D5	665	209	31.43
E1	565	198	35.04
E2	493	163	33.06
E3	434	119	27.42
E4	328	126	38.41
E5	307	109	35.50
F1	238	91	38.24
F2	179	70	39.11
F3	134	51	38.06
F4	115	53	46.09
F5	64	54	84.38
G1	73	31	42.47
G2	50	28	56.00

defaulted	0	1	ratio
sub_grade			
G3	29	19	65.52
G4	43	13	30.23
G5	20	10	50.00

In [151…

```
ct_sub.ratio.sort_values(ascending = False).plot(kind = 'bar', figsize = (60,20)
plt.show()

#from the plot it is evident that loan takers of F5 loans have the highest defau
```



RELATIONSHIP BETWEEN THE LOAN STATUS AND STATE

In [152…

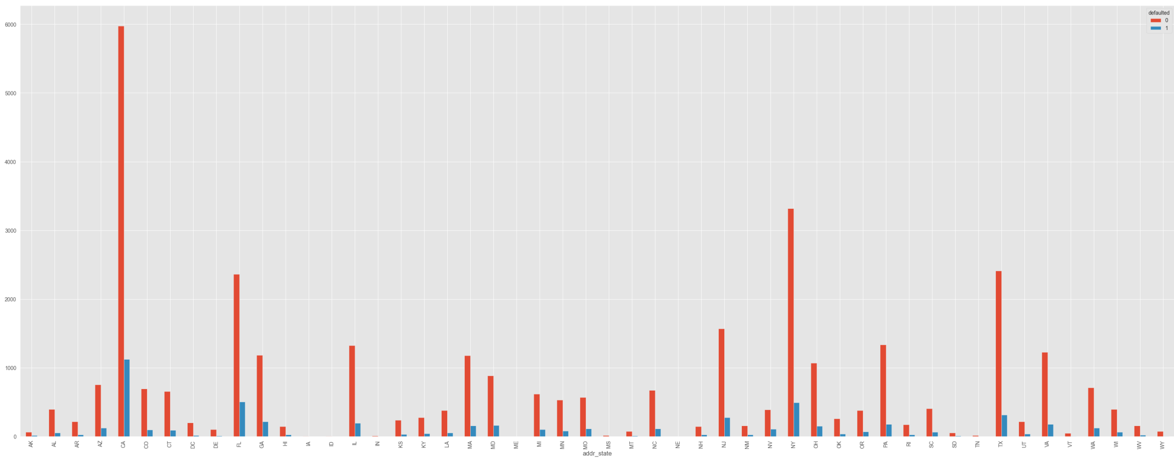
```
loan_dataset_cp.columns
```

Out[152…

```
Index(['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'term', 'int_rate',
       'installment', 'grade', 'sub_grade', 'emp_title', 'emp_length',
       'annual_inc', 'verification_status', 'issue_d', 'loan_status',
       'purpose', 'zip_code', 'addr_state', 'dti', 'inq_last_6mths',
       'open_acc', 'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
       'last_pymnt_d', 'pub_rec_bankruptcies', 'defaulted',
       'int_rate_converted', 'binned_annual_inc',
       'funded_amount_classification'],
      dtype='object')
```

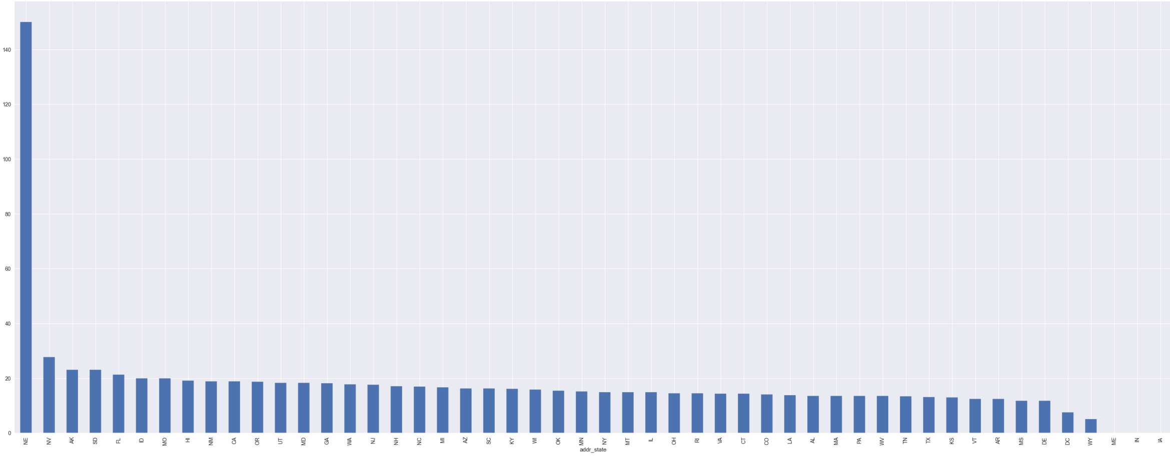
In [153…

```
ct_state = pd.crosstab(loan_dataset_cp.addr_state, loan_dataset_cp.defaulted)
ct_state.plot(kind = 'bar', figsize = (40,15))
plt.show()
```



In [154…
```
#what state have the highest default ratio.., we can see that.. let's do a reana
ct_state['ratio'] = (ct_state[1]/ct_state[0])*100
```

In [155…
```
plt.figure()
plt.style.use('seaborn')
ct_state.ratio.sort_values(ascending = False).plot(kind = 'bar', figsize = (40,1
plt.show()
```



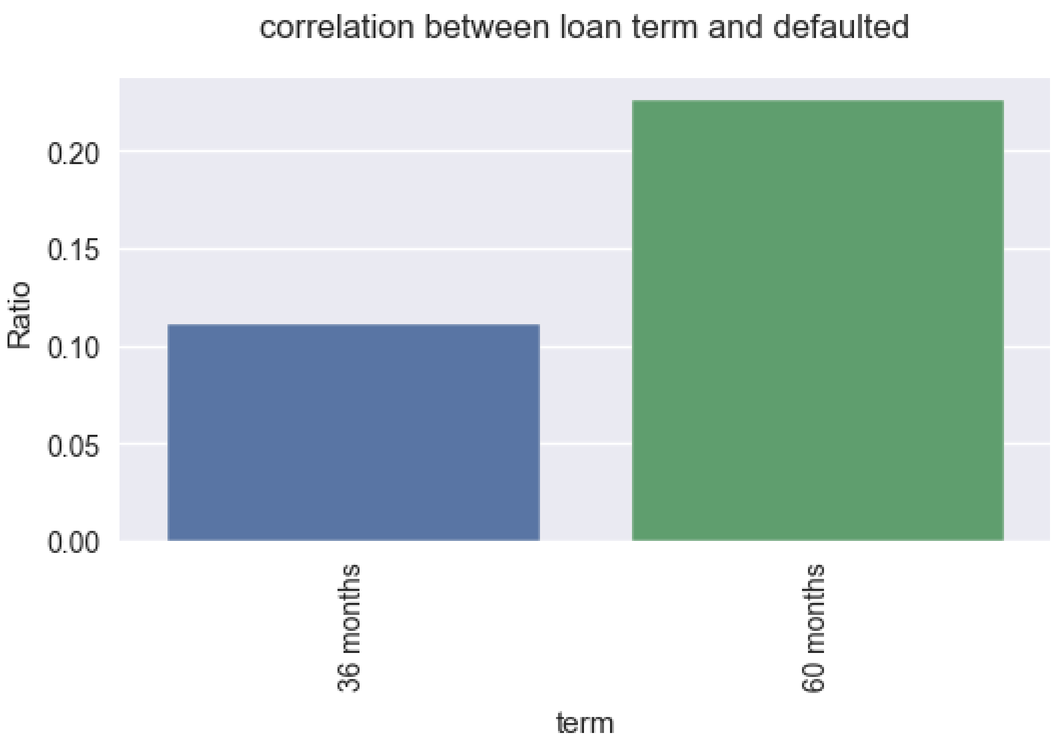
In [156…
```
ct_state.sort_values(by = 'ratio', ascending = False).head()
```

Out[156…

defaulted	0	1	ratio
addr_state			
NE	2	3	150.000000
NV	389	108	27.763496
AK	65	15	23.076923
SD	52	12	23.076923
FL	2362	504	21.337849

In [157…
```
#Let's PLot against ratio of default and the term.. i.e. the months of loan coll
```

In [158…
```
term_defaulters_df = loan_dataset_cp[loan_dataset_cp.defaulted == 1].groupby('te
term_defaulted_df = loan_dataset_cp.groupby('term')['defaulted'].count().reset_i
term_defaulted_full = pd.merge(term_defaulters_df, term_defaulted_df, on = 'term
term_defaulted_full['Ratio'] = term_defaulted_full['defaulted_yes']/term_default

plt.figure(figsize = (6,3))
sns.barplot(data = term_defaulted_full, x = 'term', y = 'Ratio')
plt.title('correlation between loan term and defaulted', pad = 15)
plt.xticks(rotation = 90)
plt.show()

#observations 60 months loans have higher defaults that 36 months loan
```



In [159…

```
#let's see if there is any relationship between the job title of the loan seeker
```

In [160…

```
loan_dataset_cp['emp_title'].value_counts().head(20)
```

Out[160…

```
emp_title
US Army                  134
Bank of America          109
IBM                       66
AT&T                      59
Kaiser Permanente         56
Wells Fargo               54
USAF                      54
UPS                       53
US Air Force              52
Walmart                   45
Lockheed Martin           44
United States Air Force   42
State of California       42
U.S. Army                 41
Verizon Wireless          40
Self Employed             40
USPS                      39
US ARMY                   39
Walgreens                 38
JP Morgan Chase           37
Name: count, dtype: int64
```

In [161…

```
loan_dataset_cp['emp_title'] = loan_dataset_cp['emp_title'].str.upper()
```

In [162…

```
#we need the list of the defaulters and the list of all the defaulted with regar
employers = loan_dataset_cp.groupby('emp_title')['defaulted'].count()
employers = employers.reset_index().sort_values('defaulted', ascending = False)
employers_defaulted = loan_dataset_cp.groupby('emp_title')['defaulted'].sum()
employers_full = pd.merge(employers, employers_defaulted, on = 'emp_title')
employers_full.rename(columns = {'defaulted_x':'Totals', 'defaulted_y':'defaulte
employers_full['Default Ratio'] = round((employers_full['defaulted']/employers_f
```

```
employers_full[employers_full.Totals > 20].sort_values(by = 'Default Ratio', asc

#a very good observation is that staffs from UPS and WALMART who had taken loan
#highest defaults
```

Out[162…

	emp_title	Totals	defaulted	Default Ratio
8	UPS	63	17	26.9841
47	WAL-MART	24	6	25.0000
2	WALMART	81	20	24.6914
37	RETIRED	33	8	24.2424
58	INTERNAL REVENUE SERVICE	21	5	23.8095
27	UNITED STATES POSTAL SERVICE	38	9	23.6842
54	US BANK	22	5	22.7273
19	US POSTAL SERVICE	45	10	22.2222
30	SELF-EMPLOYED	36	8	22.2222
50	SPRINT	23	5	21.7391
7	VERIZON WIRELESS	64	13	20.3125
3	AT&T	79	16	20.2532
25	U.S. ARMY	42	8	19.0476
51	UNITED STATES NAVY	22	4	18.1818
52	MILITARY	22	4	18.1818
10	SELF EMPLOYED	57	10	17.5439
11	USPS	57	10	17.5439
1	BANK OF AMERICA	137	24	17.5182
39	MORGAN STANLEY	29	5	17.2414
35	NORTHROP GRUMMAN	35	6	17.1429
18	HOME DEPOT	47	8	17.0213
26	TARGET	42	7	16.6667
24	JP MORGAN CHASE	43	7	16.2791
44	DEPARTMENT OF VETERANS AFFAIRS	25	4	16.0000
21	DEPARTMENT OF DEFENSE	44	7	15.9091
14	WALGREENS	53	8	15.0943
57	AMERICAN EXPRESS	21	3	14.2857
0	US ARMY	210	30	14.2857
33	BEST BUY	36	5	13.8889
29	COMCAST	37	5	13.5135
9	SELF	60	8	13.3333
5	IBM	68	9	13.2353
28	UNITED PARCEL SERVICE	38	5	13.1579

	emp_title	Totals	defaulted	Default Ratio
49	THE HOME DEPOT	23	3	13.0435
45	MERRILL LYNCH	25	3	12.0000
6	WELLS FARGO	67	8	11.9403
23	VERIZON	43	5	11.6279
4	KAISER PERMANENTE	69	8	11.5942
32	WELLS FARGO BANK	36	4	11.1111
20	UNITED STATES AIR FORCE	45	5	11.1111
40	DEPARTMENT OF HOMELAND SECURITY	27	3	11.1111
31	BOOZ ALLEN HAMILTON	36	4	11.1111
12	US AIR FORCE	57	6	10.5263
16	STATE OF CALIFORNIA	48	5	10.4167
55	US GOVERNMENT	22	2	9.0909
34	UNITED STATES ARMY	35	3	8.5714
48	RAYTHEON	24	2	8.3333
43	GENERAL ELECTRIC	26	2	7.6923
42	SOCIAL SECURITY ADMINISTRATION	27	2	7.4074
41	CITIGROUP	27	2	7.4074
17	US NAVY	47	3	6.3830
15	LOCKHEED MARTIN	49	3	6.1224
36	FIDELITY INVESTMENTS	34	2	5.8824
22	JPMORGAN CHASE	43	2	4.6512
53	COLUMBIA UNIVERSITY	22	1	4.5455
13	USAF	56	2	3.5714
38	ACCENTURE	32	0	0.0000
56	TIME WARNER CABLE	22	0	0.0000
46	PRICEWATERHOUSECOOPERS	24	0	0.0000

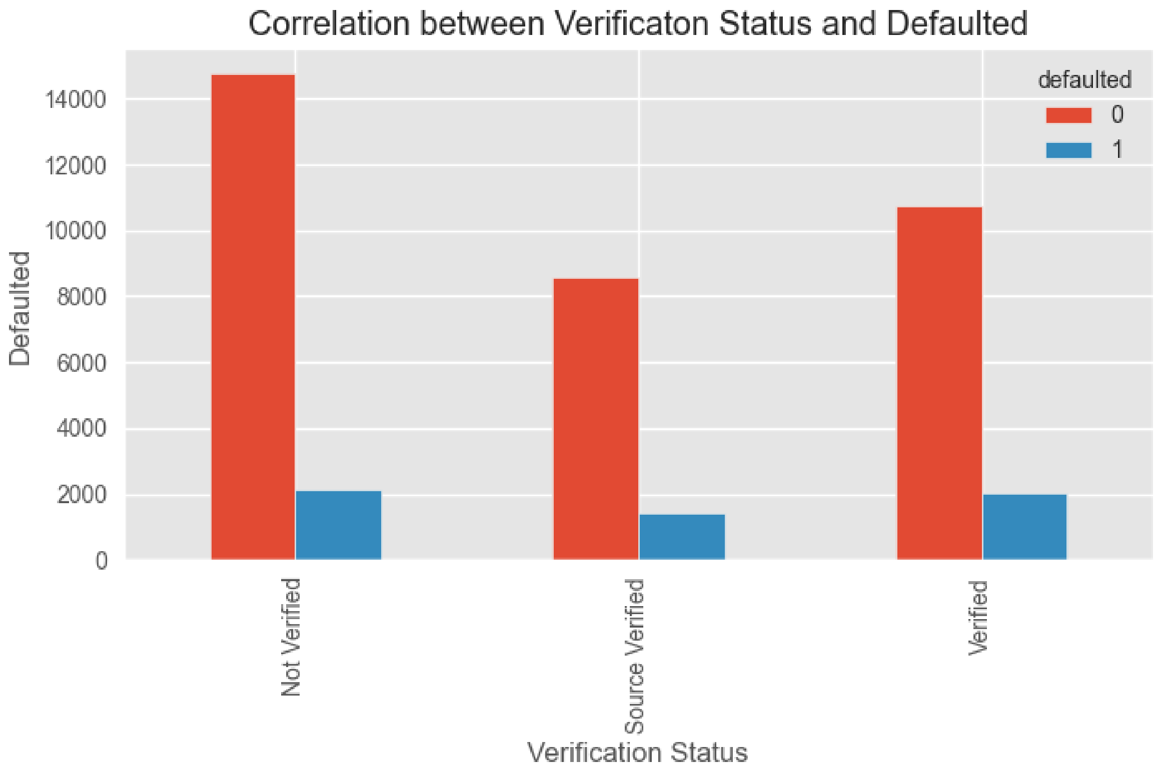
In [163…

```
#LET'S SEE THE RELATIONSHIP BETWEEN VERIFICATION STATUS AND THE DEFAULTS
```

In [164…

```
plt.figure()
plt.style.use('ggplot')
ver_def = pd.crosstab(loan_dataset_cp.verification_status, loan_dataset_cp.defau
ver_def.plot(kind = 'bar', figsize = (8,4))
plt.title('Correlation between Verificaton Status and Defaulted')
plt.xlabel('Verification Status')
plt.ylabel('Defaulted')
plt.show()
```

```
<Figure size 800x550 with 0 Axes>
```



In [165…

```
ver_def = ver_def.assign(Ratio = (ver_def[1]/(ver_def[0]+ver_def[1])*100))
```

In [166…

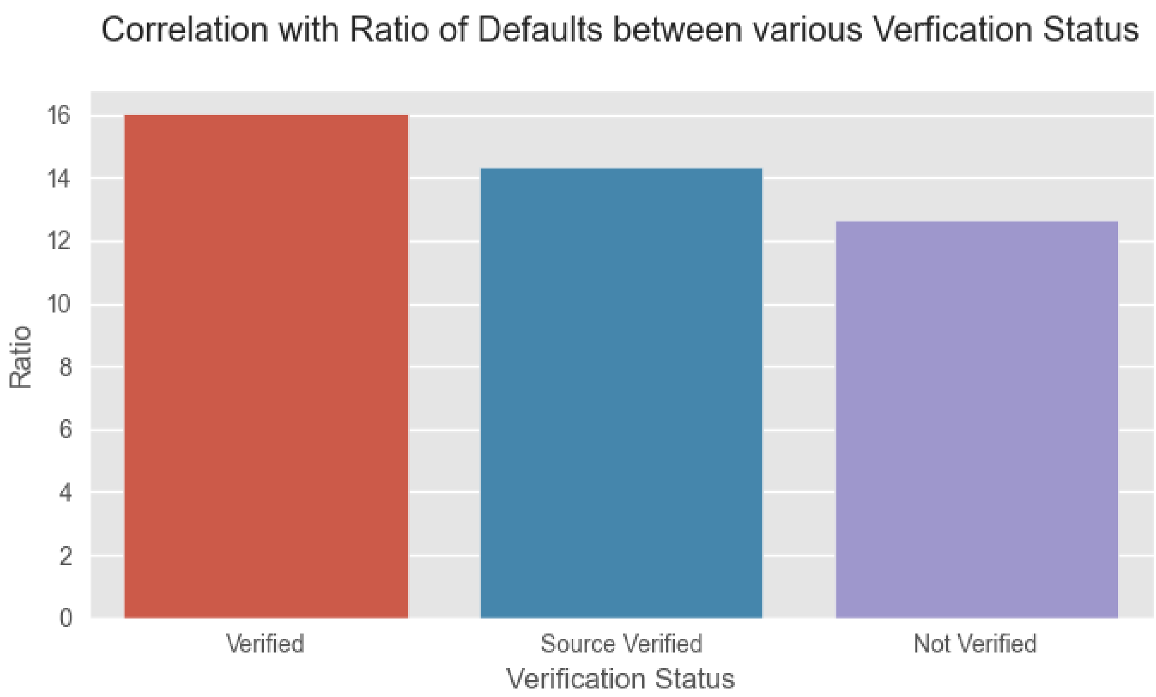
```
ver_def = ver_def.sort_values(by = 'Ratio', ascending = False)
```

In [167…

```
#let's see the visualisation
```

In [168…

```
plt.figure(figsize = (8,4))
sns.barplot(data = ver_def, x = ver_def.index, y = ver_def['Ratio'])

plt.title('Correlation with Ratio of Defaults between various Verfication Status
plt.xlabel('Verification Status')
plt.ylabel('Ratio')
plt.show()

#so from the plot, it is clear that those that were verfied defaulted more on th
```



Relationship between month name to check for a cyclical influence

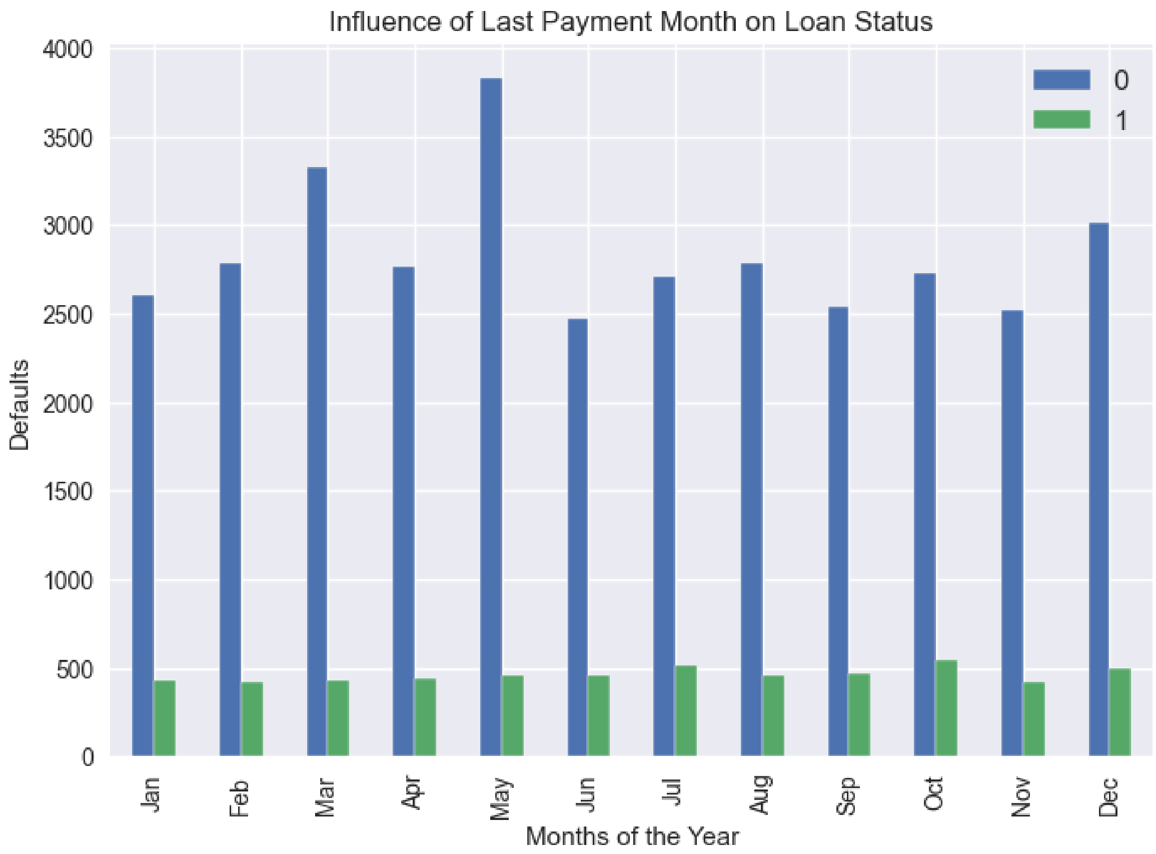
In [169…
```
from datetime import datetime
```

In [170…
```
loan_dataset_cp['last_pymnt_d'] = pd.to_datetime(loan_dataset_cp['last_pymnt_d']
```

In [171…
```
loan_dataset_cp['month_last_pymnt'] = loan_dataset_cp['last_pymnt_d'].dt.strftim
```

In [172…
```
month_names = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'C
```

In [173…
```
plt.style.use('seaborn')
pd.crosstab(loan_dataset_cp['month_last_pymnt'], loan_dataset_cp['defaulted']).r
plt.title('Influence of Last Payment Month on Loan Status')
plt.xlabel("Months of the Year")
plt.ylabel("Defaults")
plt.legend(fontsize = 12)
plt.show()

#observation is that the months of March and May have the highest loan intakes
#and the defaults were significantly low
```



CORRELATION BETWEEN LOAN STATUS AND DTI

In [174…

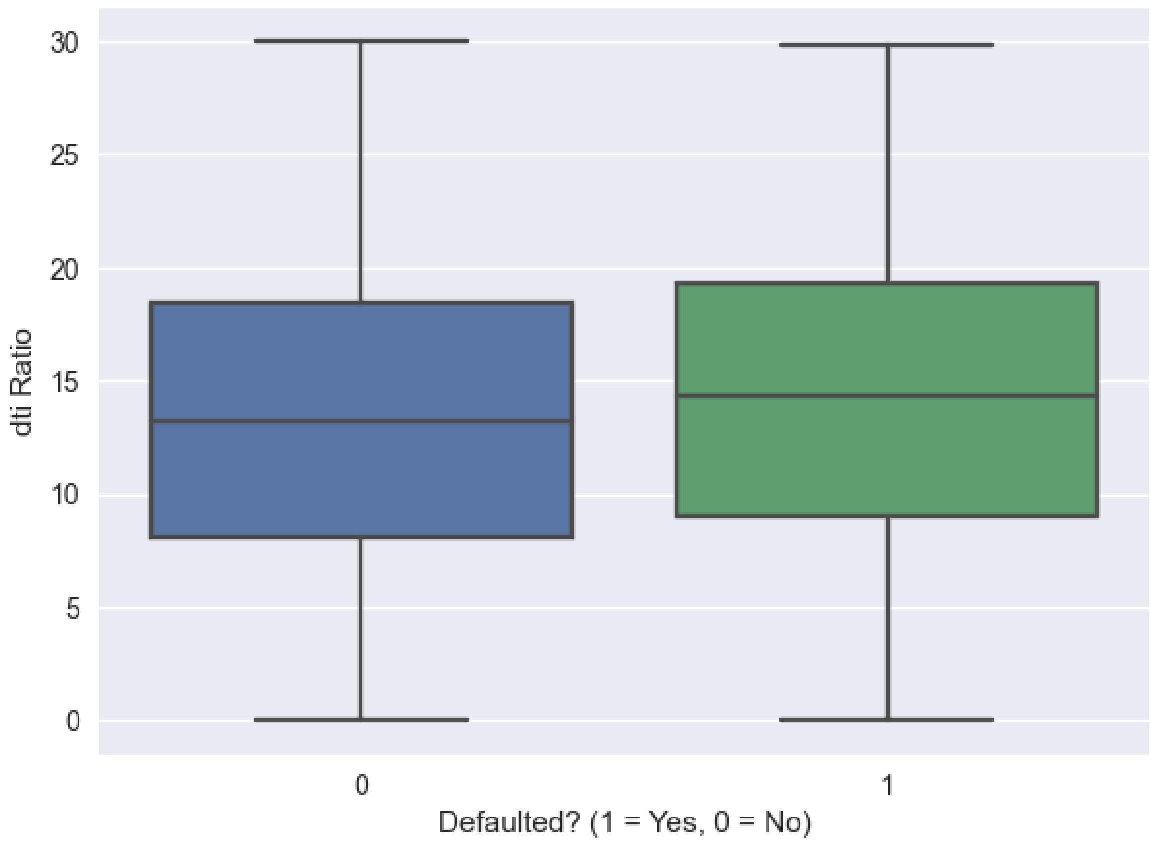
```
loan_dataset_cp.dti
```

Out[174…

```
0        27.65
1         1.00
2         8.72
3        20.00
4        17.94
         ...
39712    11.33
39713     6.40
39714     2.30
39715     3.72
39716    14.29
Name: dti, Length: 39717, dtype: float64
```

In [175…

```
#for this correlation.., let's use a boxplot.., so we will actually get to see t
fig, axes = plt.subplots(1,1)
fig.set_size_inches(7,5)
sns.boxplot(x= 'defaulted', y = 'dti', data = loan_dataset_cp, ax = axes)
axes.set_xlabel('Defaulted? (1 = Yes, 0 = No)')
axes.set_ylabel('dti Ratio')
fig.show()
```

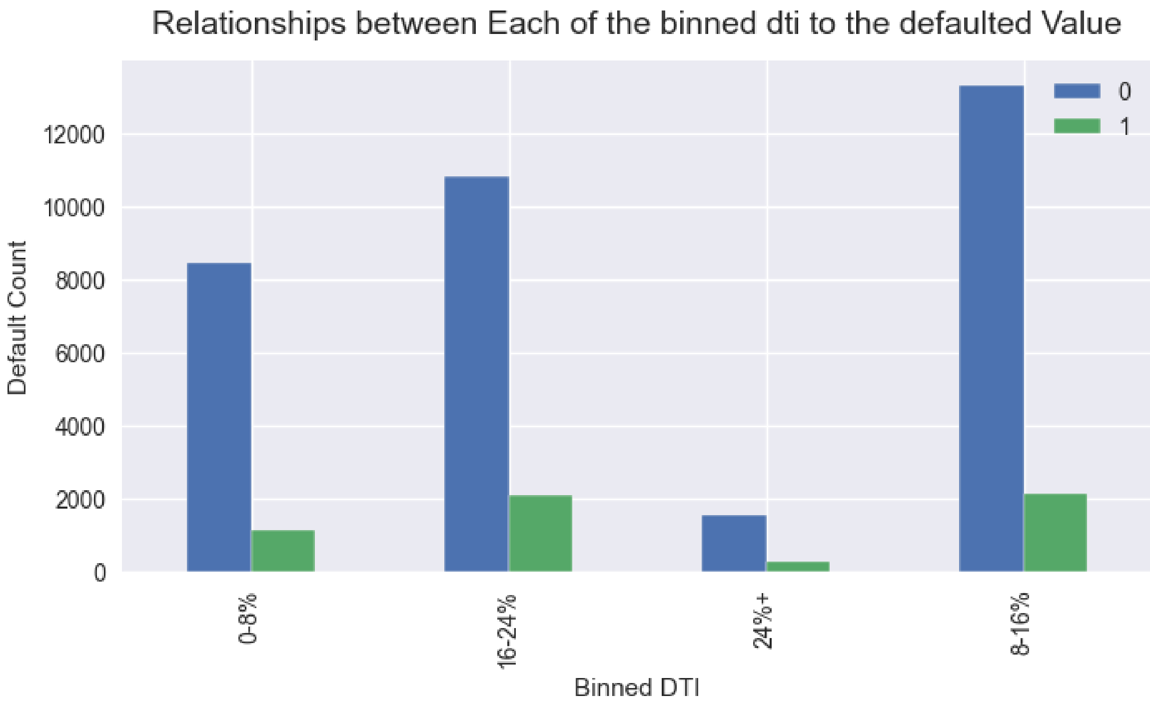


In [176…

```
#let's start by binning into [0-8%, 8-16%, 16-24%, 24%+].., this would be very h
def classify_dti(x):
    if x >= 0 and x < 8:
        return '0-8%'
    elif x >= 8 and x < 16:
        return '8-16%'
    elif x >= 16 and x < 24:
        return '16-24%'
    else:
        return '24%+'

loan_dataset_cp['dtit'] = loan_dataset_cp['dti'].apply(classify_dti)
```

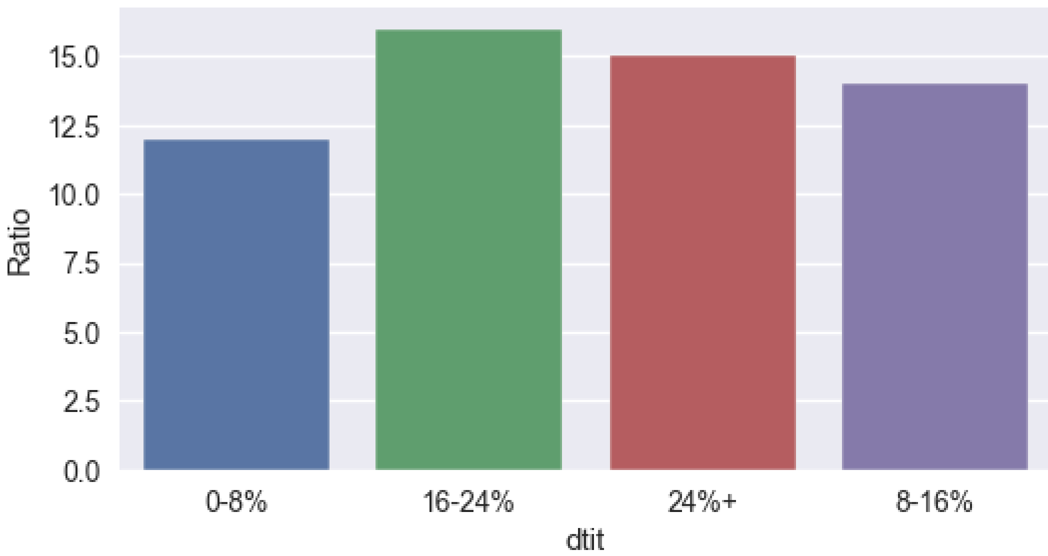
In [177…

```
dti_temp = pd.crosstab(loan_dataset_cp.dtit, loan_dataset_cp.defaulted)
dti_temp.plot(kind = 'bar', figsize = (8,4))
plt.title("Relationships between Each of the binned dti to the defaulted Value",
plt.xlabel("Binned DTI")
plt.ylabel("Default Count")
plt.legend()
plt.show()

#observation the 16-24% and the 8-16% have the highest default ratio
#but we need something more.., we need the ratio of the defaults to the non defa
```



In [178…

```
dti_temp['Ratio'] = round((dti_temp[1]/(dti_temp[0]+dti_temp[1])*100))
plt.figure(figsize = (6,3))
sns.barplot(data = dti_temp, x = dti_temp.index, y = dti_temp.Ratio)
plt.show()

#observation the dti ratio of the 16-24% has the highest default of around 18%
```

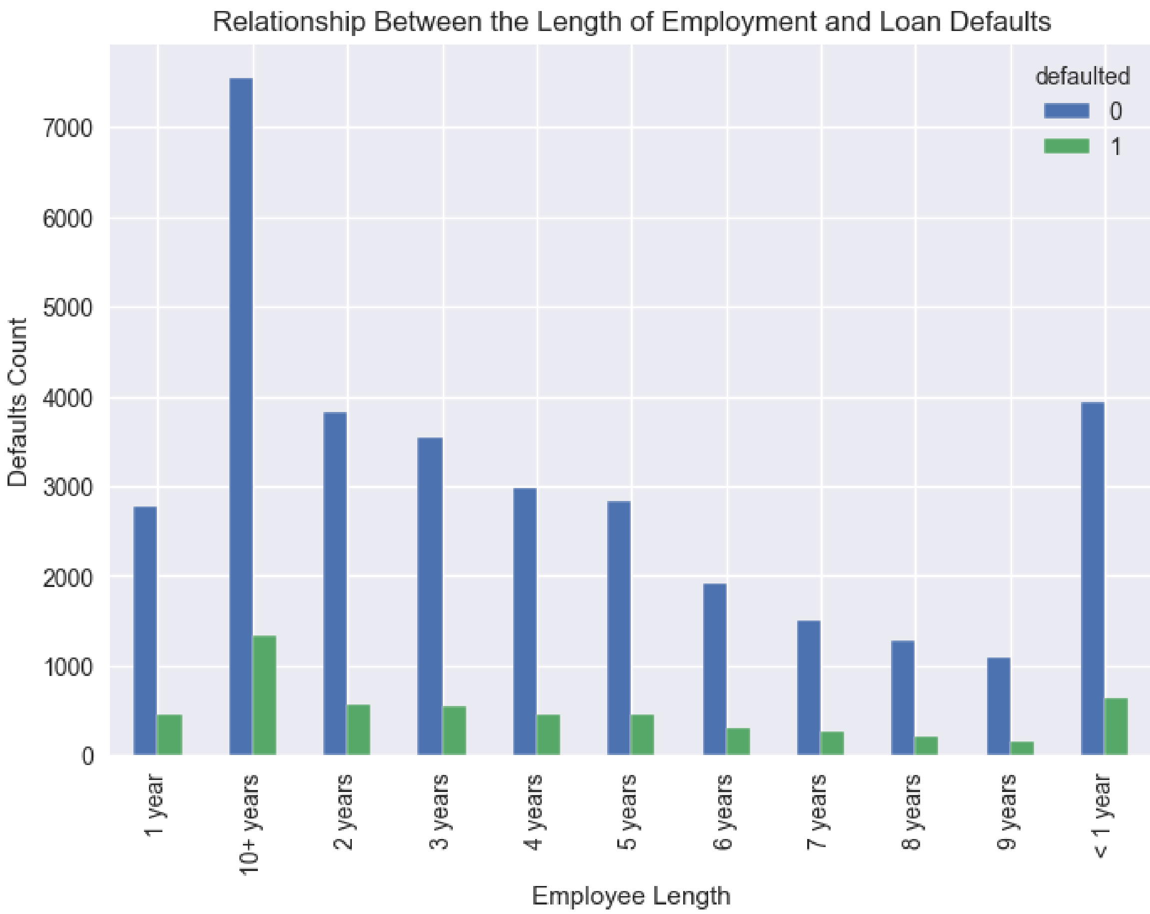


In [179…

```
##Next Up Let's find the relationship between employment tenure and Loan default
```

In [180…

```
emp_length_def = pd.crosstab(loan_dataset_cp.emp_length, loan_dataset_cp.default
emp_length_def.plot(kind = 'bar')
plt.title("Relationship Between the Length of Employment and Loan Defaults")
plt.xlabel("Employee Length")
plt.ylabel("Defaults Count")
plt.show()
```

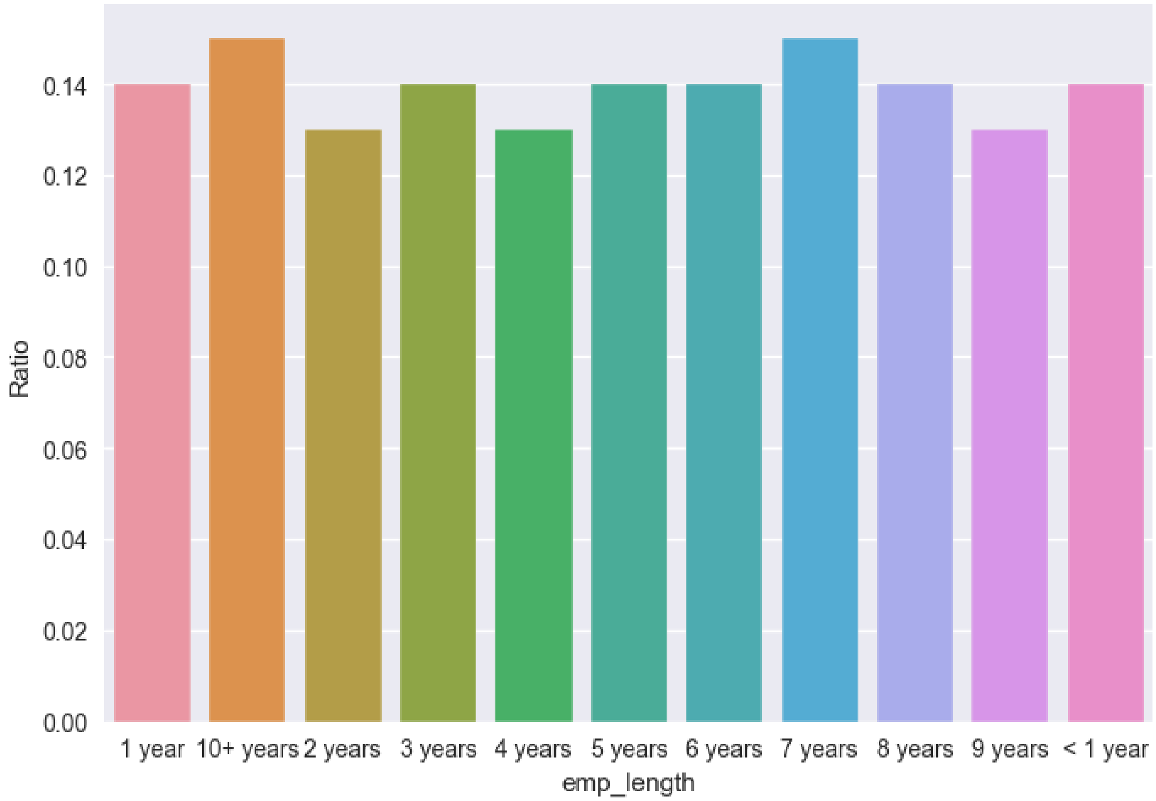


In [181…

```
#but this visualisation doesn't really give us the ratio.., so let's create anot
#bit of how the ration and plot
emp_length_def['Ratio'] = round((emp_length_def[1]/(emp_length_def[0]+emp_length
sns.barplot(data = emp_length_def, x = emp_length_def.index, y = 'Ratio')

#there is really no relationship in this case..., i.e. the length of employment
#to whether a customer would default on a loan or not
```

Out[181…

```
<Axes: xlabel='emp_length', ylabel='Ratio'>
```

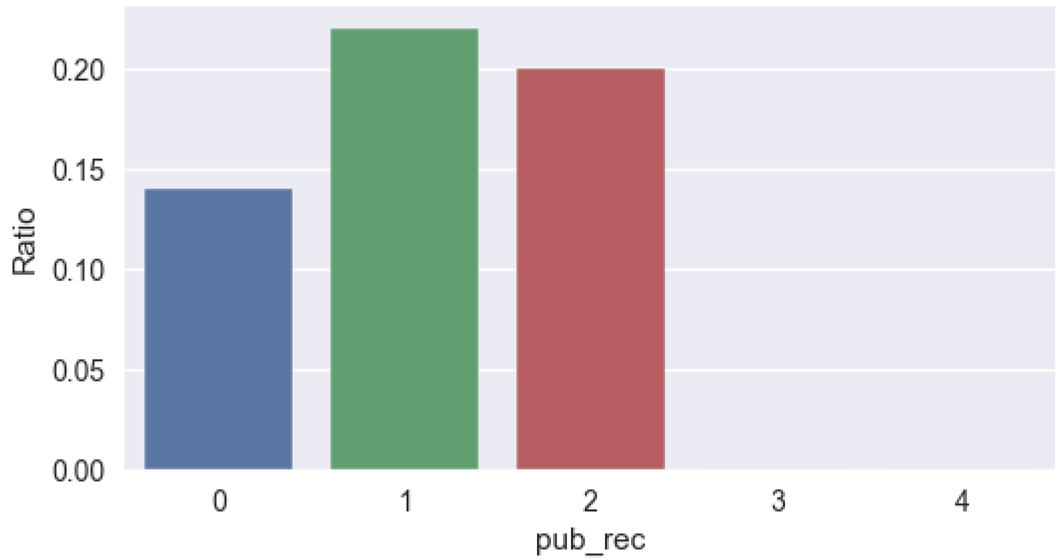


In [182…

```
#PUBLIC REC AND PUBLIC REC BANKRUPTCIES
```

In [183…

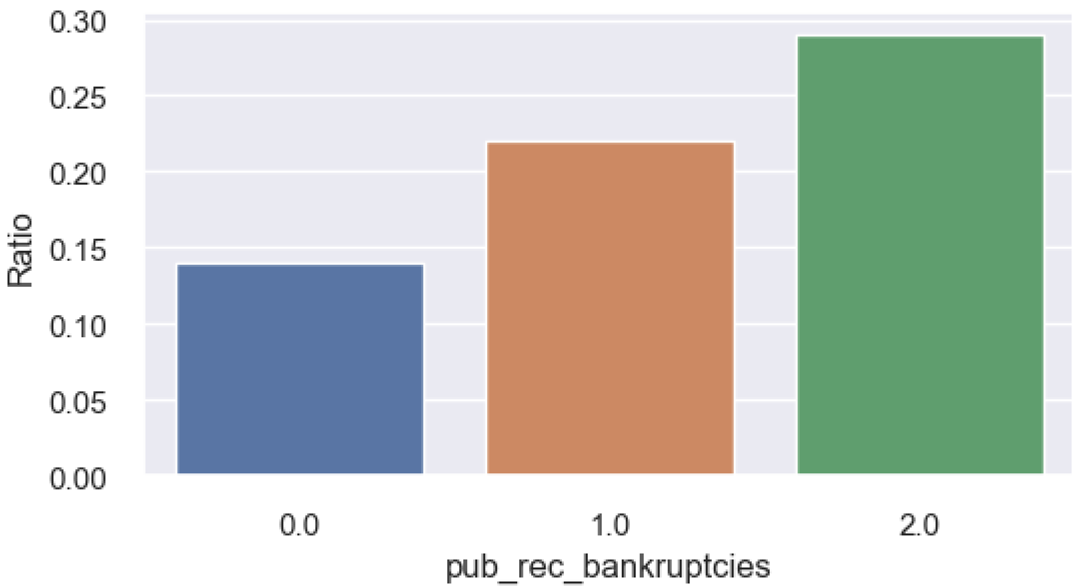
```
pub_rec_def = pd.crosstab(loan_dataset_cp.pub_rec, loan_dataset_cp.defaulted)
pub_rec_def['Ratio'] = round(pub_rec_def[1]/(pub_rec_def[1] + pub_rec_def[0]), 2
plt.figure(figsize = (6,3))
sns.barplot(data = pub_rec_def, x = pub_rec_def.index, y = pub_rec_def.Ratio)
sns.set(font_scale = 1)
plt.show()

#A pub_rec of 1 has a very high default ratio.., 2 is very high too almost reach
```



In [184…

```
#Let's do the same visualisations for public_#rec_bankruptcies and see the outco
#alright.. alright alright.. let's proceed and see more on the bankruptcies

pub_rec_bnkt_def = pd.crosstab(loan_dataset_cp['pub_rec_bankruptcies'], loan_dat
pub_rec_bnkt_def['Ratio'] = round(pub_rec_bnkt_def[1]/(pub_rec_bnkt_def[1] + pub
```

```
plt.figure(figsize = (6,3))
sns.barplot(data = pub_rec_bnkt_def, x = pub_rec_bnkt_def.index, y = pub_rec_bnk
sns.set(font_scale = 1)
plt.show()

#important observation: borrowers with pub_rec_bankruptcies of 2 have a higher p
```

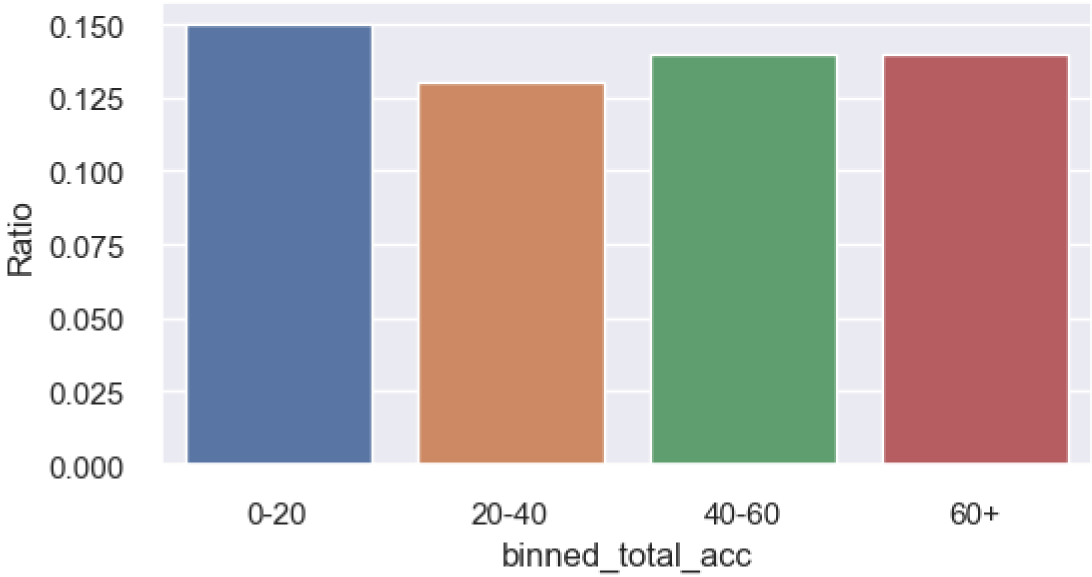


In [185…

```
#Let's check if there is any relationship between tacc and defaults
```

In [186…

```
#because the dataset is numbers of different category..., it would be beneficial
#0-20, 20-40, 40-60, 60+ and then plot

def classify_total_acc(x):
    if x >= 0 and x < 20:
        return '0-20'
    elif x >=20 and x < 40:
        return '20-40'
    elif x >= 40 and x < 60:
        return '40-60'
    else:
        return '60+'

loan_dataset_cp['binned_total_acc'] = loan_dataset_cp['total_acc'].apply(classif

total_acc_def = pd.crosstab(loan_dataset_cp['binned_total_acc'], loan_dataset_cp
total_acc_def['Ratio'] = round(total_acc_def[1]/(total_acc_def[1] + total_acc_de
plt.figure(figsize = (6,3))
sns.barplot(data = total_acc_def, x = total_acc_def.index, y = total_acc_def.Rat
sns.set(font_scale = 1)
plt.show()

#observation: there is really no relationship between the total_acc and the defa
```



In [187…

```
#let's go over some rough work and understand
#some very important intricates
```

In [188…

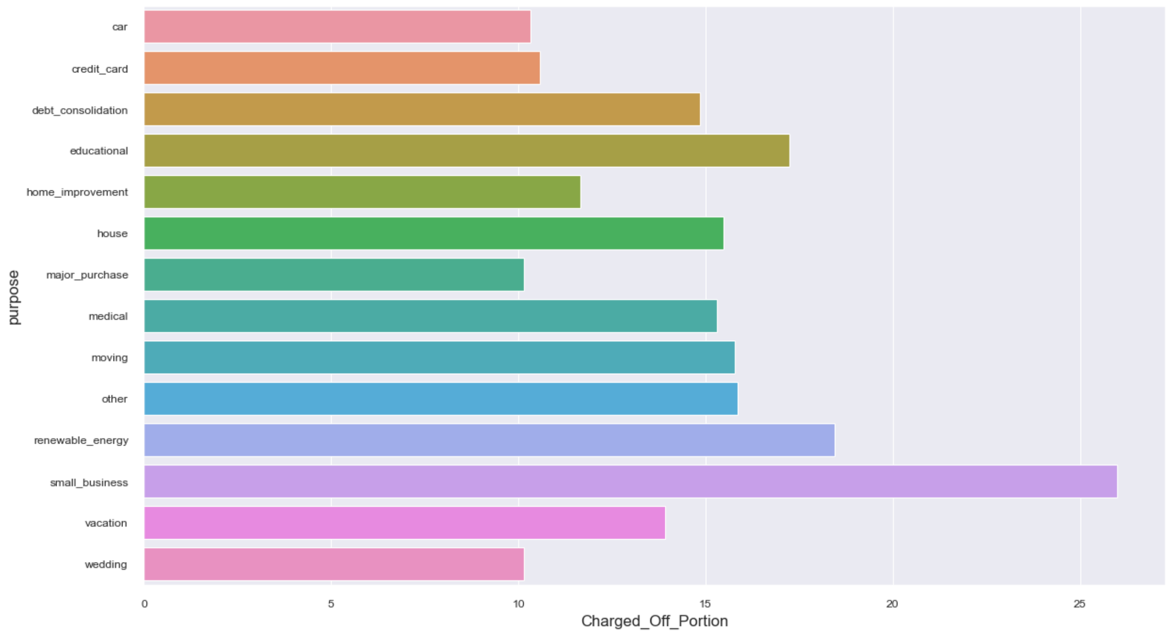
```
purpose_vs_loan = loan_dataset_cp.groupby(['purpose', 'loan_status'])['loan_stat
#we also want the total of all the loan statuses for each purpose
purpose_vs_loan['Total'] = purpose_vs_loan['Charged Off'] + purpose_vs_loan['Cur
#we also want the purpose vs loan that is teh charged off portion from the total
purpose_vs_loan['Charged_Off_Portion'] = (purpose_vs_loan['Charged Off']/purpose
purpose_vs_loan.sort_values(by = 'Charged_Off_Portion', ascending = False)

#so from this table, we can conclude that small business applicants have higher
```

Out[188…

loan_status	purpose	Charged Off	Current	Fully Paid	Total	Charged_Off_Portio
11	small_business	475.0	74.0	1279.0	1828.0	25.98468
10	renewable_energy	19.0	1.0	83.0	103.0	18.44660
3	educational	56.0	0.0	269.0	325.0	17.23076
9	other	633.0	128.0	3232.0	3993.0	15.85274
8	moving	92.0	7.0	484.0	583.0	15.78044
5	house	59.0	14.0	308.0	381.0	15.48556
7	medical	106.0	12.0	575.0	693.0	15.29581
2	debt_consolidation	2767.0	586.0	15288.0	18641.0	14.84362
12	vacation	53.0	6.0	322.0	381.0	13.91076
4	home_improvement	347.0	101.0	2528.0	2976.0	11.65994
1	credit_card	542.0	103.0	4485.0	5130.0	10.56530
0	car	160.0	50.0	1339.0	1549.0	10.32924
6	major_purchase	222.0	37.0	1928.0	2187.0	10.15089
13	wedding	96.0	21.0	830.0	947.0	10.13727

In [189…

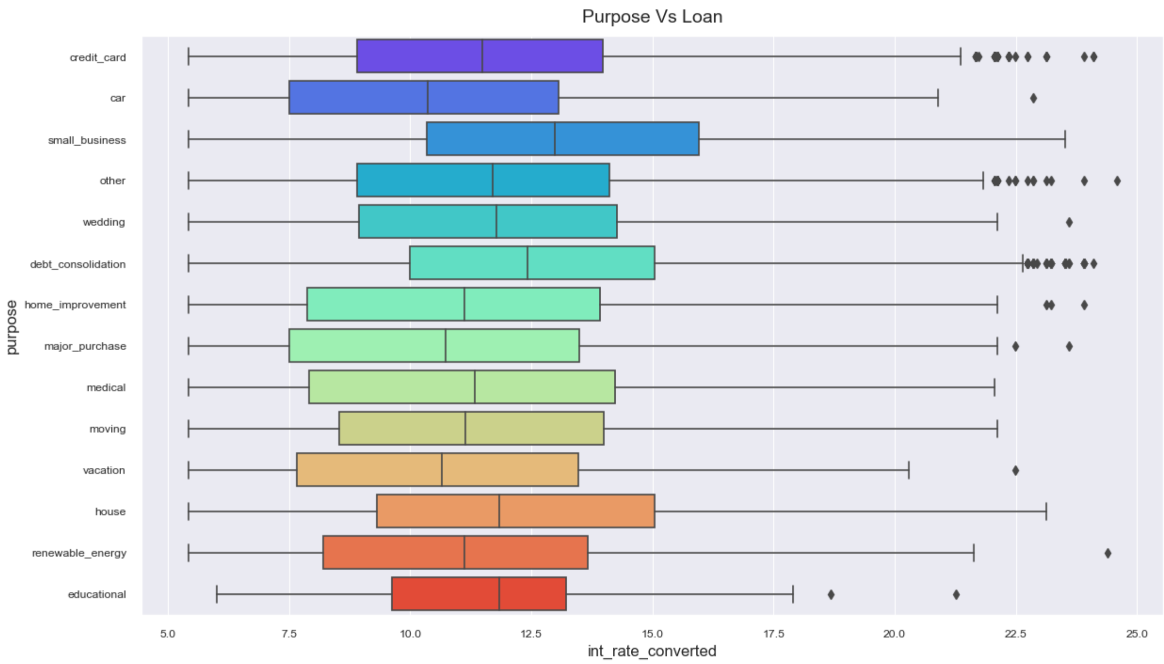
```
#Let's get the visualisation of the charged off portion and the purpose

sns.set_context("paper", rc = {"font.size":12, "axes.titlesize":12, "axes.labels
fig, ax1 = plt.subplots(figsize = (14,8))
ax1 = sns.barplot(y = 'purpose', x = 'Charged_Off_Portion', data = purpose_vs_lo
fig.show()
```



In [190…

```
##Let's see something else
#Bivariate Analysis: Purpose Vs Interest Rate
```

```
sns.set_context('paper', rc = {'font.size':12, 'axes.titlesize':12, 'axes.labels
fig = plt.figure(figsize = (14,8))
ax1 = fig.add_subplot(111)
ax1 = sns.boxplot(x = 'int_rate_converted', y = 'purpose', data = loan_dataset_c
ax1.set_title('Purpose Vs Loan', fontsize = 14, pad = 10)
fig.show()

#observation: for small businesses we charge high interest rate i.e.loans taken
#debt consolidation have a whole lot of outliers, some people are charged very h
#both other and debt consolidation have many outliers that might have eventually
```

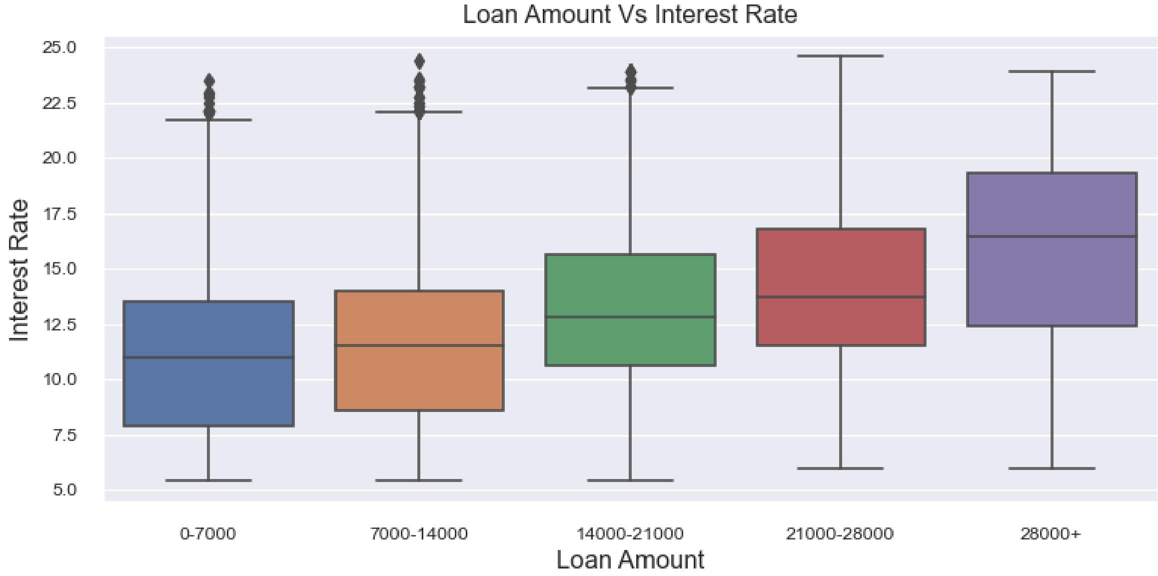


In [191…

```
## Let's see if there is any relationship between the loan amount and the Intere
```

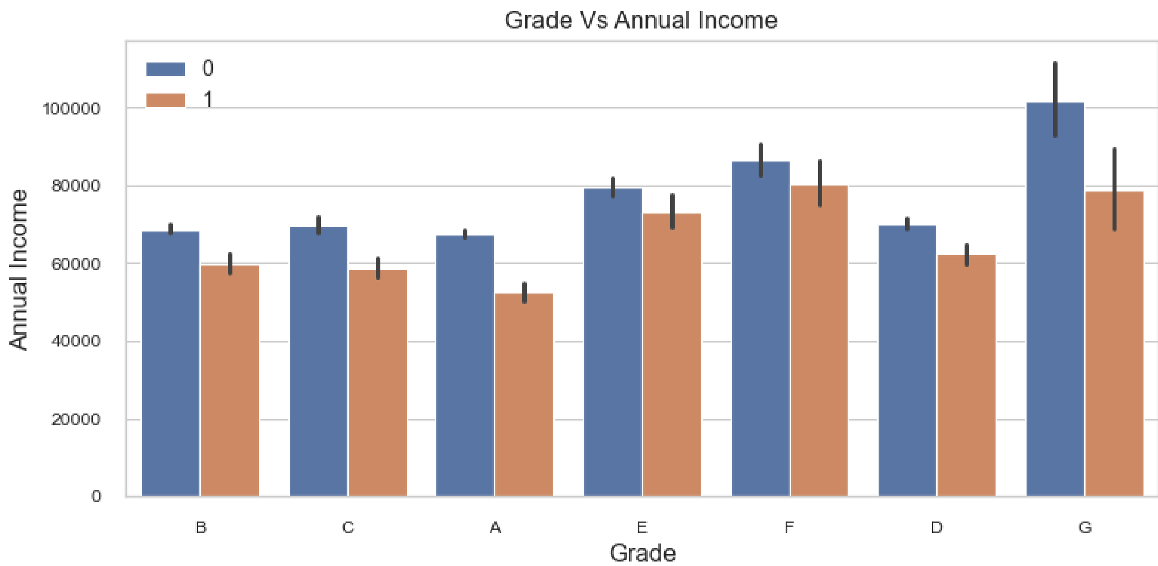
In [192…

```
loan_dataset_cp['loan_amnt_cat'] = pd.cut(loan_dataset_cp['loan_amnt'], [0, 7000



sns.set_context('paper', rc = {'font.size':12, 'axes.titlesize':12, 'axes.labels
plt.figure(figsize = (9,4))
ax1 = fig.add_subplot(111)
ax1 = sns.boxplot(x = 'loan_amnt_cat', y = 'int_rate_converted', data = loan_dat
ax1.set_title("Loan Amount Vs Interest Rate")
ax1.set_xlabel("Loan Amount")
ax1.set_ylabel("Interest Rate")
plt.show()

#quick observation: the higher the loan amount, the higher the interest rate
```



In [193…

```
#Bivariate Analysis : To show the correlation between the grades of loan and The
sns.set_context('paper', rc = {'font.size':12, 'axes.titlesize':12, 'axes.labels
plt.figure(figsize = (9,4))
sns.set_style('whitegrid')
ax1 = fig.add_subplot(111)
ax1 = sns.barplot(x = 'grade', y = 'annual_inc', data = loan_dataset_cp, hue = '
ax1.set_title("Grade Vs Annual Income")
ax1.set_xlabel("Grade")
ax1.set_ylabel("Annual Income")
plt.legend(fontsize = 10)
plt.show()

#Observations?
#people getting charged off have much lover average annual incomes
```



Find out the Correlation Between Interest Rate and Default

In [194…

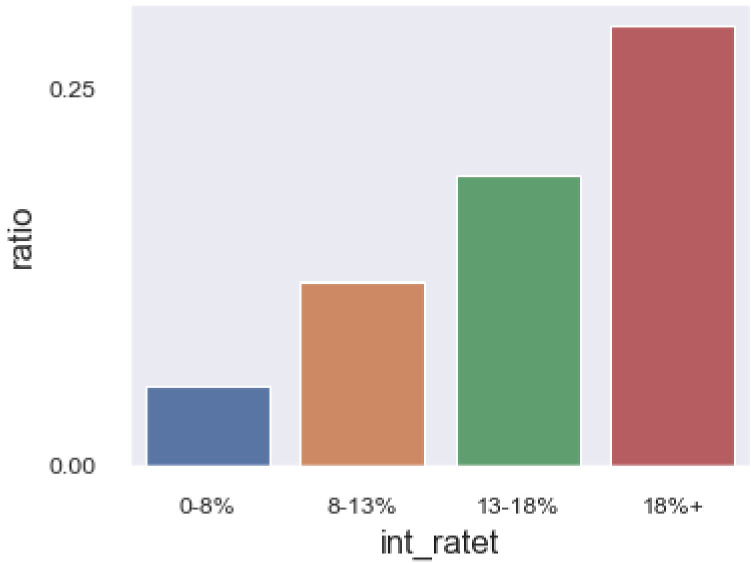
```
loan_dataset_cp['int_ratet'] = pd.cut(loan_dataset_cp['int_rate_converted'], [0,
```

In [195…

```
loan_dataset_cp['int_ratet']
```

Out[195…
```
0         8-13%
1        13-18%
2        13-18%
3        13-18%
4         8-13%
          ...
39712     8-13%
39713     8-13%
39714     8-13%
39715      0-8%
39716    13-18%
Name: int_ratet, Length: 39717, dtype: category
Categories (4, object): ['0-8%' < '8-13%' < '13-18%' < '18%+']
```

In [196…
```
## Now, let's visualise this by plotting between the interest rate categories an
int_ratet_def = pd.crosstab(loan_dataset_cp.int_ratet, loan_dataset_cp.defaulted
int_ratet_def['ratio'] = (int_ratet_def[1]/(int_ratet_def[1] + int_ratet_def[0])

sns.set_context('paper', rc = {'font.size':14, 'axes.titlesize':12, 'axes.labels
plt.figure(figsize = (4,3))
sns.set_style('dark')
sns.barplot(data = int_ratet_def, x = 'int_ratet', y = 'ratio')
sns.set(font_scale = 11)
plt.show()

#observation the higher the interest rate.., the more likely is it for someone t
```



CONCLUSION

Observations:

The Following Observations were made after the EDA of the Lending Club Loan Dataset

- Plotting the Annual Incomes: Took a 99% of the Annual Income that removed High outliers and showed meaningful comparison (similar thing was done for 95% of the Annual Income as well.., and produced even better results). It does show a tendency towards more Defaults by people having lesser incomes. Annual income do have a degree of -ve correlation with default rate. 0 - 30K income bracket show a significantly high default rate. Default rate of people in income bracket of 0 - 30K taking loans >20K is extremely high. - ~50%. Even, Default rate of people in income bracket of 0 - 30K taking loans 10K - 20K is high - ~25%. These combinations must be removed

- Grade/Sub-Grade: Products with certain grade/sub-grade combinations lead to high to very high defaults and should be looked at. (i.e. F and G)
- State: Applicants from Nebraska hasa very High Default Ratio - 60% but in a very small population. Alaska and South Dakota observe moderately high default ratio - ~18%
- Purpose: "Small Businesss" purpose loands tends to show a very high default trend. Other than that, Renewable energy loans have a moderately high default rate. Avoid requests with these types.
- Funded Amount: has a very high +ve correlation with Defaulted. Loans > 20K have significantly higher default rate than loans of 0 - 10K. Installment amount have a +ve correlation with Defaulted
- Interest Rate Amount: has a very high +ve correlation with Defaulted. Loans with 18% and higher rate show a much higher default rate. Consider lower rate loans more
- From the Plotting of dti Ratio: there is an upward trend of Defaults as the dti ratio goes higher. Focusing on customers with lower dti ratios is better for bringing down Default Ratio.
- Pub_rec_bankruptcies: High +ve correlation of Default to number of pub_rec_bankrupticies. Avoid customers with any bankruptcy record
- Pub_rec: Customers with any derogatory public record has more propensity to default. Avoid customers with derogatory record
- Term: 60 months tenured loans show a much higher default rate. Avoid higher tenured loans.
- Emp_title: (UPS): An analysis was carried out to find out companies from which 20 employees have taken credit from lending club and has high default rate. These companies are worth being careful about

NEXT STEP:

- Let's export to a csv format
- and we can prepare a dashboard in power bi to highlight these important findings

```
In [ ]:
```