Lending Case Study EDA

In [60]:

```
#Load the libraries which will be required further for analysis
import pandas as pd #To work with dataset
import numpy as np #Math library
import seaborn as sns #Graph library that use matplot in background
import matplotlib.pyplot as plt #to plot some parameters in seaborn
import warnings
import calendar
warnings.filterwarnings("ignore")
```

In [61]:

```
#Load the data and print few rows, provide the path before executing the notebook.
lending_case = pd.read_csv("....../lending-case-study/loan.csv")
lending_case.head(2)
```

Out[61]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83

2 rows × 111 columns

In [62]:

```
#Check number of rows and columns before performing data cleaning
print("Rows and column :: ", lending_case.shape)
```

```
Rows and column ::  (39717, 111)
```

Data Cleaning (Summary)

- Fix rows and columns i.e. remove columns containing NA
- Fix missing values
- Standardise values i.e. fix the right data type
- Filter Data (For ex: loan approved amount can not be greater than Loan applied amount)
- Drop Duplicates
- Result: Rows dropped from 39717 to 37868
- Result: Columns dropped from 111 to 44

Check the columns with all the values as NA and drop them

In [63]:

```
print("Shape before dropping columns  :: ", lending_case.shape)
columns_to_drop = lending_case.columns[lending_case.isna().all()].tolist()
lending_case = lending_case.drop(columns = columns_to_drop)
print("Shape after dropping columns with all missing values :: ", lending_case.shape
```

```
Shape before dropping columns  ::  (39717, 111)
Shape after dropping columns with all missing values ::  (39717, 57)
```

Analyse on missing values by checking column values, if they contains null

In [64]:

```
#Function to print data frame stats where columns has missing value greater than zer
def print_columns_with_missing_value(df_table):
    missing_values_column = df_table.isnull().sum().to_frame('missing_values')
    print(missing_values_column[missing_values_column.missing_values > 0])

print_columns_with_missing_value(lending_case)
```

```
                            missing_values
emp_title                             2459
emp_length                            1075
desc                                 12940
title                                   11
mths_since_last_delinq               25682
mths_since_last_record               36931
revol_util                              50
last_pymnt_d                            71
next_pymnt_d                         38577
last_credit_pull_d                       2
collections_12_mths_ex_med              56
chargeoff_within_12_mths                56
pub_rec_bankruptcies                   697
tax_liens                               39
```

Drop column with missing rows and and not required for identifying pattern for new loan

In [65]:

```
#drop columns
columns_to_drop = ['emp_title', 'mths_since_last_delinq', 'mths_since_last_record',
lending_case = lending_case.drop(columns = columns_to_drop)
print("Shape after dropping additional columns :: ", lending_case.shape)

#drop columns with no imp. Note: tax_liens
columns_with_no_imp = ['member_id', 'url', 'desc', 'zip_code', 'tax_liens']
lending_case = lending_case.drop(columns = columns_with_no_imp)
print("Shape after dropping columns with no importance :: ", lending_case.shape)
```

```
Shape after dropping additional columns ::  (39717, 53)
Shape after dropping columns with no importance ::  (39717, 48)
```

In [66]:

```
#check again columns with missing values
print_columns_with_missing_value(lending_case)
```

```
                             missing_values
emp_length                             1075
title                                    11
revol_util                               50
last_pymnt_d                             71
last_credit_pull_d                        2
collections_12_mths_ex_med               56
chargeoff_within_12_mths                 56
pub_rec_bankruptcies                    697
```

In [67]:

```
#Drop additional columns as its not needed, our purpose is to flag the customer wher
additional_colums_to_drop = ['collections_12_mths_ex_med', 'chargeoff_within_12_mths
lending_case = lending_case.drop(columns = additional_colums_to_drop)
print("Shape after dropping columns with no importance :: ", lending_case.shape)
```

```
Shape after dropping columns with no importance ::  (39717, 42)
```

Data Imputation fill unknown column values

In [68]:

```
#Replace employee length as zero i.e. they do not have an experience
lending_case['emp_length'].unique() #unique values
lending_case['emp_length'].fillna('0', inplace = True)
lending_case['emp_length'].isnull().sum()

#Fill the missing values Unknown, this parameter may be used in combination with oth
lending_case['pub_rec_bankruptcies'].unique()
lending_case['pub_rec_bankruptcies'].fillna('Unknown', inplace = True)
lending_case['pub_rec_bankruptcies'].isnull().sum()
```

Out[68]:

```
0
```

In [69]:

```
#check columns with missing values again
print_columns_with_missing_value(lending_case)
```

```
       missing_values
title              11
```

Analyze the column title and replace with appropriate values

Most frequent title is Debt Consolidataion

In [70]:

```
#print(lending_case['title'].value_counts())
lending_case['title'].fillna('Debt Consolidation', inplace = True)
print_columns_with_missing_value(lending_case)
```

```
Empty DataFrame
Columns: [missing_values]
Index: []
```

Analyze rows with missing values (summary)

No rows were found with missing values more than 2/3

In [117]:

```
missing_column_values_for_row = lending_case.isnull().sum(axis=1).to_frame('missing_
print(missing_column_values_for_row[missing_column_values_for_row.missing_values > 2
```

```
Empty DataFrame
Columns: [missing_values]
Index: []
```

Check the data types of columns to be analysed

Our focus should be on column shown as object, which dataframe was unable to determine right data type

In [118]:

```
#Analyze columsn with data type 'object' and fix data type
lending_case.dtypes
```

Out[118]:

```
id                               int64
loan_amnt                        int64
term                            object
int_rate                       float64
installment                    float64
grade                           object
sub_grade                       object
emp_length                       int64
home_ownership                  object
annual_inc                     float64
verification_status             object
issue_d                 datetime64[ns]
loan_status                     object
pymnt_plan                      object
purpose                         object
title                           object
addr_state                      object
dti                            float64
delinq_2yrs                      int64
earliest_cr_line                object
inq_last_6mths                   int64
open_acc                         int64
pub_rec                          int64
revol_bal                        int64
total_acc                        int64
initial_list_status             object
out_prncp                      float64
out_prncp_inv                  float64
total_pymnt                    float64
total_pymnt_inv                float64
total_rec_prncp                float64
total_rec_int                  float64
total_rec_late_fee             float64
recoveries                     float64
collection_recovery_fee        float64
last_pymnt_amnt                float64
application_type                object
pub_rec_bankruptcies            object
issue_d_year                     int64
issue_d_month                   object
annual_inc_range              category
int_rate_range                category
loan_amnt_range               category
dti_range                     category
dtype: object
```

In [72]:

```
#analyze the columns, glance what data they have before changing
print(lending_case.columns)
lending_case.head(2)
```

```
Index(['id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'term',
'int_rate',
       'installment', 'grade', 'sub_grade', 'emp_length', 'home_owner
ship',
       'annual_inc', 'verification_status', 'issue_d', 'loan_status',
       'pymnt_plan', 'purpose', 'title', 'addr_state', 'dti', 'delinq
_2yrs',
       'earliest_cr_line', 'inq_last_6mths', 'open_acc', 'pub_rec',
       'revol_bal', '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_amnt', 'policy_code',
       'application_type', 'acc_now_delinq', 'pub_rec_bankruptcies'],
      dtype='object')
```

Out[72]:

Fix int_rate and emp_length data type

In [73]:

```
lending_case['int_rate'] = lending_case['int_rate'].str.replace('[%]', '')
lending_case['emp_length'] = lending_case['emp_length'].str.replace('[^0-9]+', '')

columns_to_float_dtype = ['int_rate', 'emp_length']
lending_case[columns_to_float_dtype] = lending_case[columns_to_float_dtype].apply(pd

#Print columns data types and confirm if they have right data type
lending_case.dtypes
```

Out[73]:

```
id                           int64
loan_amnt                    int64
funded_amnt                  int64
funded_amnt_inv            float64
term                        object
int_rate                   float64
installment                float64
grade                       object
sub_grade                   object
emp_length                   int64
home_ownership              object
annual_inc                 float64
verification_status         object
issue_d                     object
loan_status                 object
pymnt_plan                  object
purpose                     object
title                       object
addr_state                  object
dti                        float64
delinq_2yrs                  int64
earliest_cr_line            object
inq_last_6mths               int64
open_acc                     int64
pub_rec                      int64
revol_bal                    int64
total_acc                    int64
initial_list_status         object
out_prncp                  float64
out_prncp_inv              float64
total_pymnt                float64
total_pymnt_inv            float64
total_rec_prncp            float64
total_rec_int              float64
total_rec_late_fee         float64
recoveries                 float64
collection_recovery_fee    float64
last_pymnt_amnt            float64
policy_code                  int64
application_type            object
acc_now_delinq               int64
pub_rec_bankruptcies        object
dtype: object
```

Derive the columns for analysis

Extract the year and month as it may be beneficial when the users are applying

In [74]:

```
#pd.to_datetime(lending_case['issue_d'], format="%b-%y").dt.year
lending_case['issue_d'] = pd.to_datetime(lending_case['issue_d'], format="%b-%y")
lending_case['issue_d_year'] = lending_case['issue_d'].dt.year
lending_case['issue_d_month'] = lending_case['issue_d'].dt.month.apply(lambda x: cal
```

In [75]:

```
print(lending_case[['issue_d', 'issue_d_year', 'issue_d_month']].head(2))
print(lending_case['issue_d'].dtypes)
```

```
     issue_d  issue_d_year issue_d_month
0 2011-12-01          2011           Dec
1 2011-12-01          2011           Dec
datetime64[ns]
```

Data Filtering

Filter the rows where funded_amnt is greater than loan_amnt and funded_amnt_inv < funded_amnt

In [76]:

```
### Safety check for ex. loan_amnt < funded_amnt and funded_amnt < funded_amnt_inv
print("length before filtering ::", len(lending_case))
lending_case = lending_case[(lending_case['loan_amnt'] <= lending_case['funded_amnt'
print("length after filtering ::", len(lending_case))
```

```
length before filtering :: 39717
length after filtering :: 37868
```

Drop duplicates

Remove the duplicates

In [77]:

```
lending_case.drop_duplicates()
print("length after dropping duplicates ::", len(lending_case))
```

```
length after dropping duplicates :: 37868
```

Univariate analysis (summary)

1. Loan status : Out of all the loans 14% are defaulted
2. Purpose in defaulted loan : debit_consolidation, credit_card, other, small_business & home improvement contributes more than anyone else
3. Home ownership : Rent and Mortgage contributes more in defaulted loans
4. Interest rate : Approx 0.2% are in outlier
5. Annual income: 10% are in outlier
6. Loan amount : Loan & funded amount are similar funded amount can be ignored later
7. Funded amount : Will be ignored as distribution is similar to loan & amount

8. issue_d & derived variable analysis: Loans are issued more in dec month

In [78]:

```
#print columns before analyzing the values
lending_case.columns
```

Out[78]:

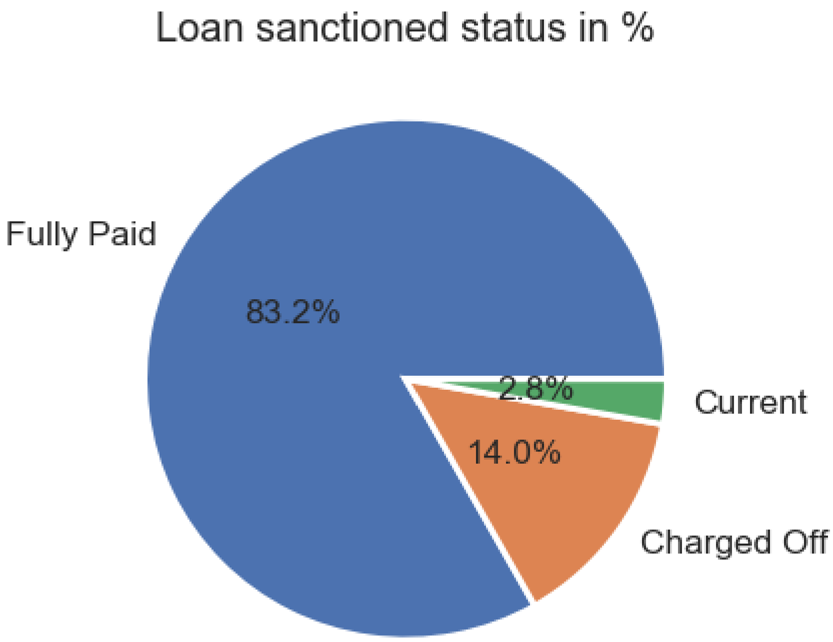
```
Index(['id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'term', 'i
nt_rate',
       'installment', 'grade', 'sub_grade', 'emp_length', 'home_owners
hip',
       'annual_inc', 'verification_status', 'issue_d', 'loan_status',
       'pymnt_plan', 'purpose', 'title', 'addr_state', 'dti', 'delinq_
2yrs',
       'earliest_cr_line', 'inq_last_6mths', 'open_acc', 'pub_rec',
       'revol_bal', 'total_acc', 'initial_list_status', 'out_prncp',
       'out_prncp_inv', 'total_pymnt', 'total_pymnt_inv', 'total_rec_p
rncp',
       'total_rec_int', 'total_rec_late_fee', 'recoveries',
       'collection_recovery_fee', 'last_pymnt_amnt', 'policy_code',
       'application_type', 'acc_now_delinq', 'pub_rec_bankruptcies',
       'issue_d_year', 'issue_d_month'],
      dtype='object')
```

Categorical variable loan_status analysis

In [79]:

```
# pie chart function
def print_pie_chart (title, pie_chart_df):
    pie, ax = plt.subplots(figsize=[10,6])
    labels  = pie_chart_df.keys()
    plt.pie(x = pie_chart_df, autopct="%.1f%%", labels=labels, pctdistance=0.5,
            wedgeprops={'linewidth': 3.0, 'edgecolor': 'white'}, textprops={'size':
    plt.title(title, fontsize=20)

#Check the loan status i.e. how many loans were paid
loan_status_pie_chart = round((lending_case['loan_status'].value_counts()/len(lendir
print_pie_chart('Loan sanctioned status in %', loan_status_pie_chart)
```



Categorical variable "Purpose" analysis who defaulted

Observation: debit_consolidation, credit_card, other, small_business & home improvementcontributes more than anyone else

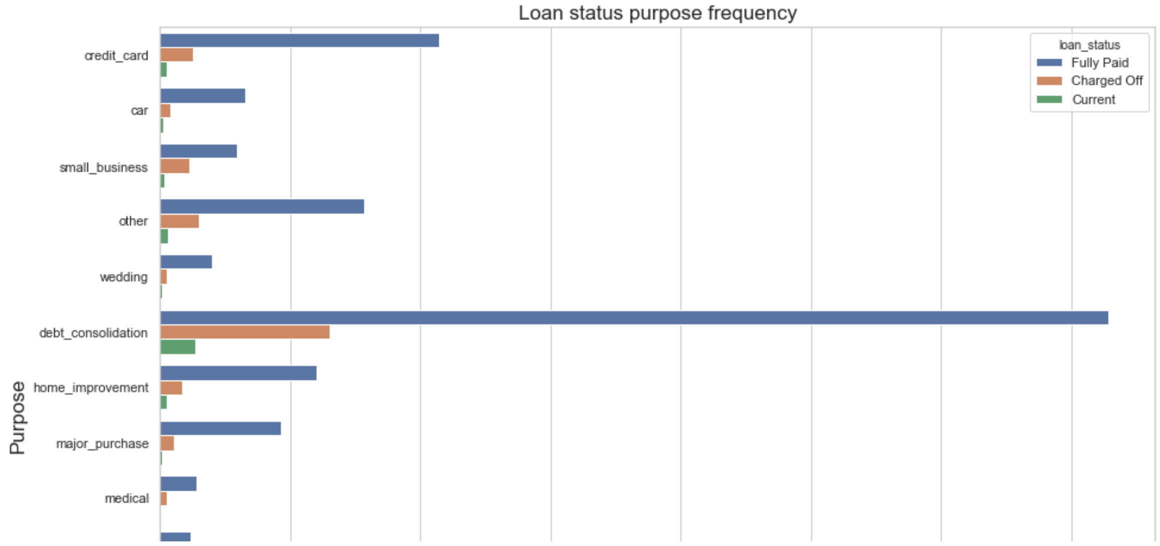
In [80]:

```
plt.figure(figsize=[15,12])
ax = sns.countplot(y = "purpose", hue="loan_status", data = lending_case)

ax.set_title("Loan status purpose frequency", fontsize=17)
ax.set_xlabel("Frequency", fontsize=17)
ax.set_ylabel("Purpose", fontsize=17)
```

Out[80]:

```
Text(0, 0.5, 'Purpose')
```



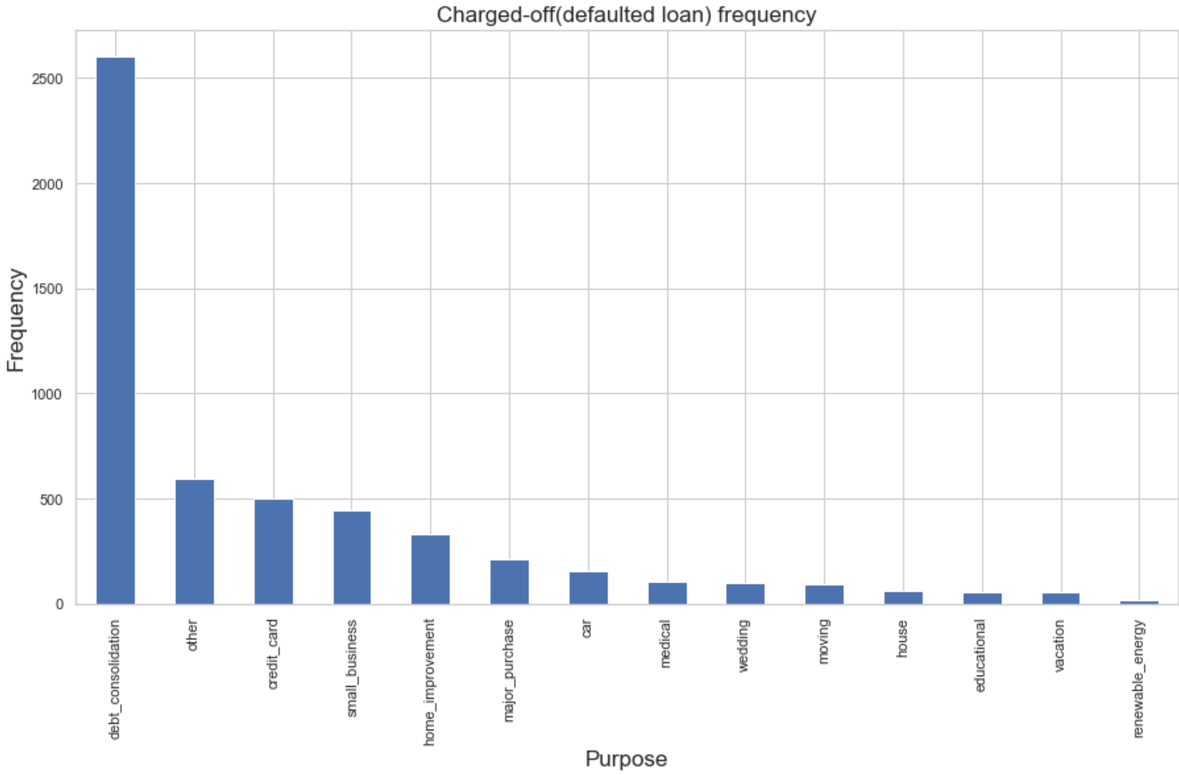
In [81]:

```
# Further Analysis for only defaulted loans
plt.figure(figsize=[15,8])
charged_off_loans = lending_case[(lending_case['loan_status'] == 'Charged Off')]
ax = charged_off_loans['purpose'].value_counts().plot.bar()

ax.set_title("Charged-off(defaulted loan) frequency", fontsize=17)
ax.set_xlabel("Purpose", fontsize=17)
ax.set_ylabel("Frequency", fontsize=17)
```

Out[81]:

```
Text(0, 0.5, 'Frequency')
```



Categocial variable "Home ownership" analysis

Observation: RENT & MORTAGE contributes alot for defaulted loans

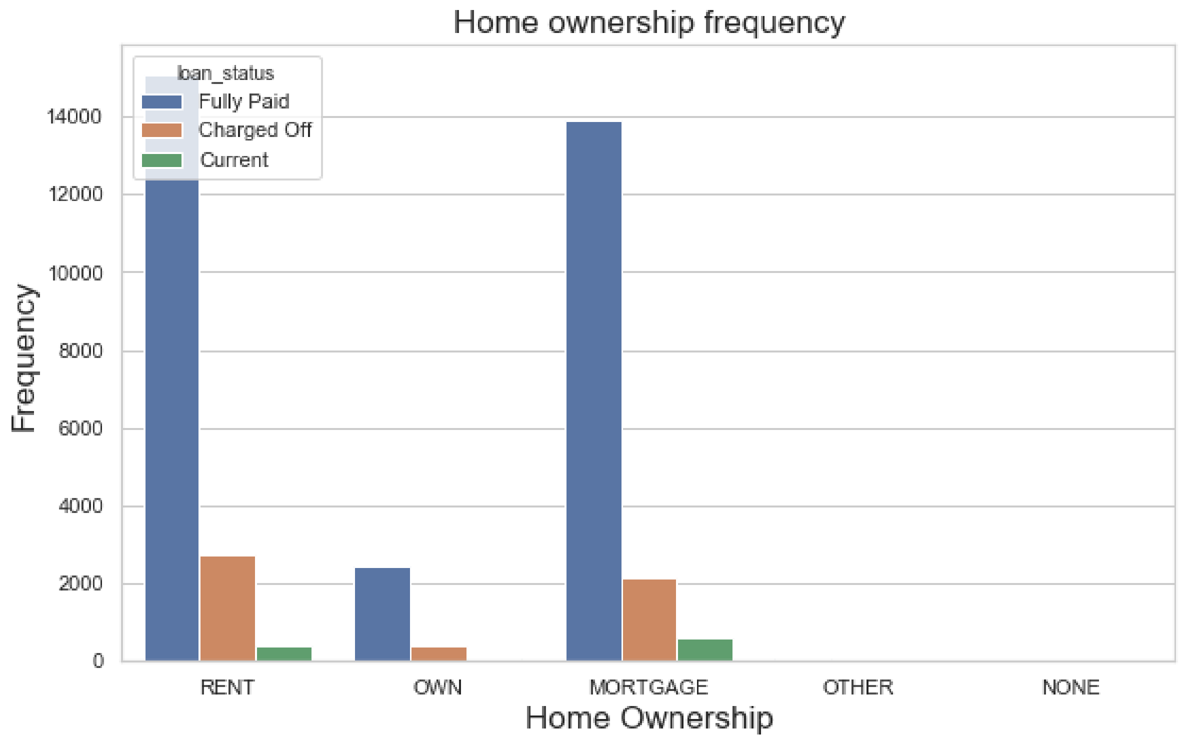
In [82]:

```
#print(lending_case.groupby(['home_ownership'])['loan_status'].value_counts())
plt.figure(figsize=[10,6])
ax = sns.countplot(x = "home_ownership", hue="loan_status", data =lending_case)

ax.set_title("Home ownership frequency", fontsize=17)
ax.set_xlabel("Home Ownership", fontsize=17)
ax.set_ylabel("Frequency", fontsize=17)
```

Out[82]:

```
Text(0, 0.5, 'Frequency')
```



Analysis of quantifier variables.

- loan_amount: funded_amnt will be after approving the loan but before analyzing check the avg.
- interest_rate: analysis box blot
- annual_inc: annual income analysis

- tenure: As the loan applied for specific duration adn unique() function shows only two lets skip it for now

Interest rate analysis

- We can ignore outliers, use quantile to ignore the percentage column
- Approx 0.1% values are in outliers, Keeping the values and rows will be filtered later if required after careful analysing all the graph

In [83]:

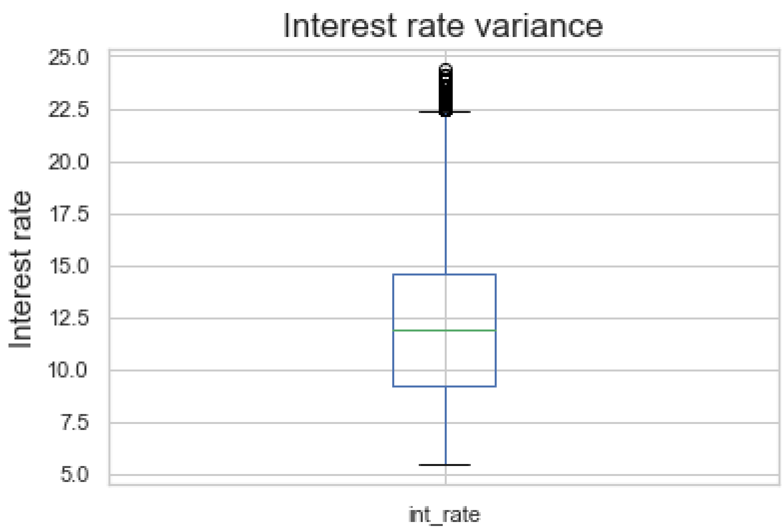
```
print(lending_case['int_rate'].describe())
ax = lending_case['int_rate'].plot.box()
ax.set_title("Interest rate variance", fontsize=17)
ax.set_ylabel("Interest rate", fontsize=15)

#99% of people are getting interest rate < 20.00 , lending_case['int_rate'].quantile
```

```
count    37868.000000
mean        12.015777
std          3.693572
min          5.420000
25%          9.250000
50%         11.860000
75%         14.540000
max         24.400000
Name: int_rate, dtype: float64
```

Out[83]:

```
Text(0, 0.5, 'Interest rate')
```



loan_amount analysis

- Approx 0.5% values are in outliers, Keeping the values and rows will be filtered later if required

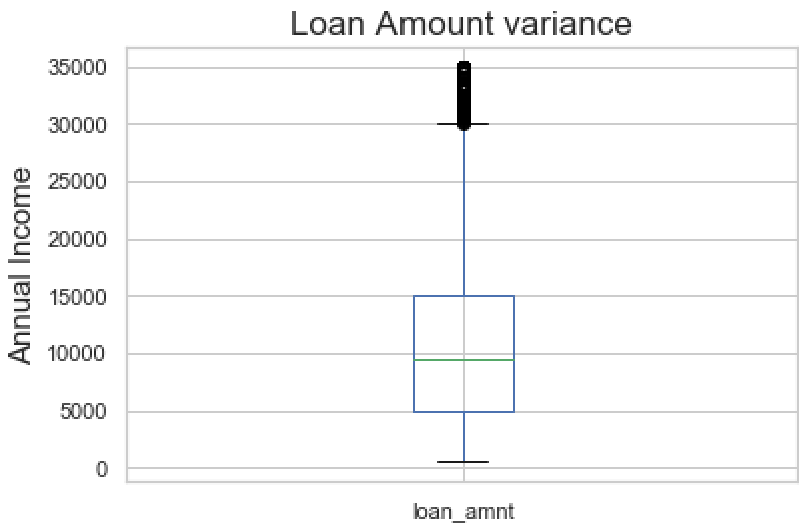
In [84]:

```
print(lending_case['loan_amnt'].describe())
ax = lending_case['loan_amnt'].plot.box()
ax.set_title("Loan Amount variance", fontsize=17)
ax.set_ylabel("Annual Income", fontsize=15)
#95% of people are less than within limit <= 25000.0 lending_case['loan_amnt'].quant
```

```
count    37868.000000
mean     10844.408207
std       7229.445777
min        500.000000
25%       5000.000000
50%       9500.000000
75%      15000.000000
max      35000.000000
Name: loan_amnt, dtype: float64
```

Out[84]:

```
Text(0, 0.5, 'Annual Income')
```



Annual_income analysis

- Almost 10% of the annual incomes are fall in outlier
- In Bi-variate analysis it was observed that filtering the rows based on outliers has no impact

In [85]:

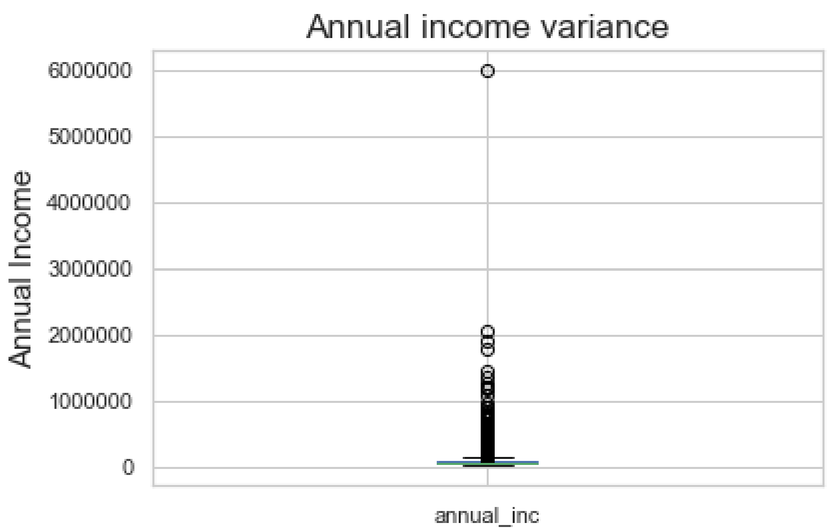
```
print(lending_case['annual_inc'].describe())
ax = lending_case['annual_inc'].plot.box()

ax.set_title("Annual income variance", fontsize=17)
ax.set_ylabel("Annual Income", fontsize=15)
```

```
count    3.786800e+04
mean     6.821043e+04
std      6.121633e+04
min      4.000000e+03
25%      4.000000e+04
50%      5.800000e+04
75%      8.100000e+04
max      6.000000e+06
Name: annual_inc, dtype: float64
```

Out[85]:

```
Text(0, 0.5, 'Annual Income')
```



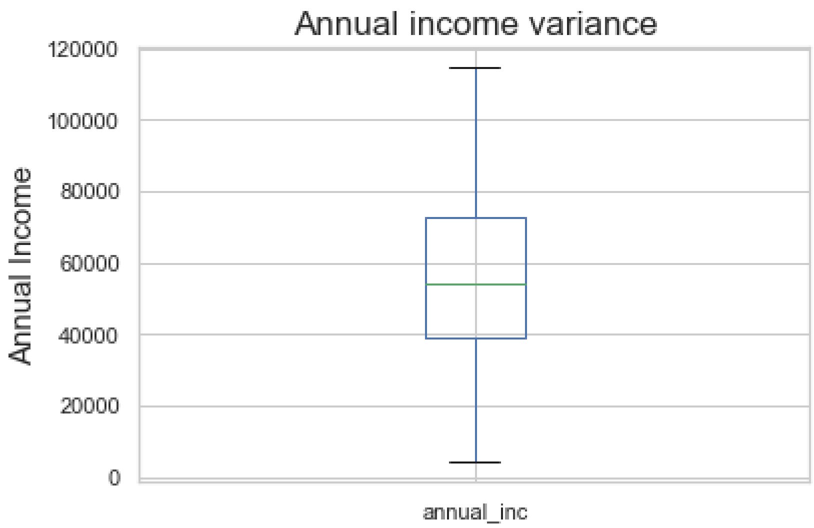
In [86]:

```
#90% of the people are not in outliers, only top 10% of the people are, we can ignor
ax = lending_case[(lending_case['annual_inc'] < lending_case['annual_inc'].quantile(

ax.set_title("Annual income variance", fontsize=17)
ax.set_ylabel("Annual Income", fontsize=15)
```

Out[86]:

```
Text(0, 0.5, 'Annual Income')
```



Loan amount, funded_amnt & funded_amnt_inv analysis

Distributions are same it means only one of them can be used, we can use loan_amnt for our analysis

In [87]:

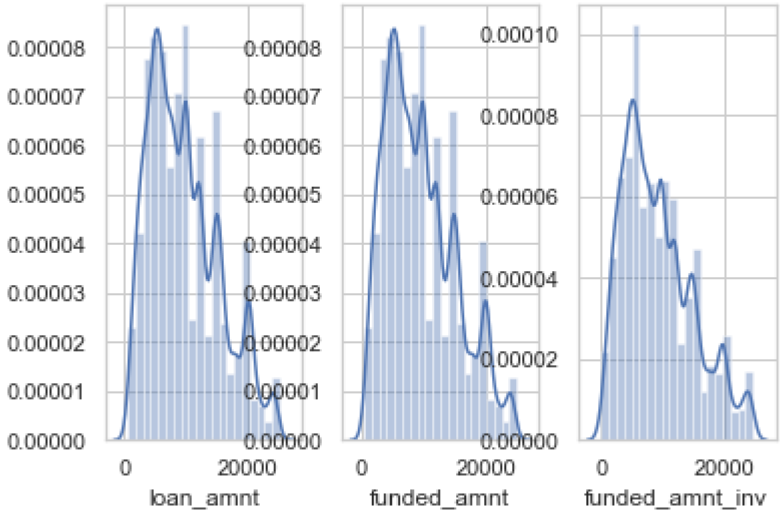
```
plt.figure(figsize=[15,6])
fig, ax =plt.subplots(1,3)
loan_amnt = lending_case[(lending_case['loan_amnt'] < lending_case['loan_amnt'].quar
sns.distplot(loan_amnt['loan_amnt'], bins = 20, ax=ax[0])

funded_amnt = lending_case[(lending_case['funded_amnt'] < lending_case['funded_amnt'
sns.distplot(funded_amnt['funded_amnt'], bins = 20, ax=ax[1])

funded_amnt_inv = lending_case[(lending_case['funded_amnt_inv'] < lending_case['func
sns.distplot(funded_amnt_inv['funded_amnt_inv'], bins = 20, ax=ax[2])

plt.show()
```

```
<Figure size 1080x432 with 0 Axes>
```



In [88]:

```
#drop columns funded_amnt & funded_amnt_inv as loan_amnt is alone sufficient for pre
columns_not_required_analysis = ['funded_amnt', 'funded_amnt_inv']
lending_case = lending_case.drop(columns = columns_not_required_analysis)
print("Shape after dropping columns with no importance :: ", lending_case.shape)
```

```
Shape after dropping columns with no importance ::  (37868, 42)
```

Loan paying term analysis (Summary)

Charged off loans are same in both 36 & 60 months, and number of Full Paid applicants are more in 36 months

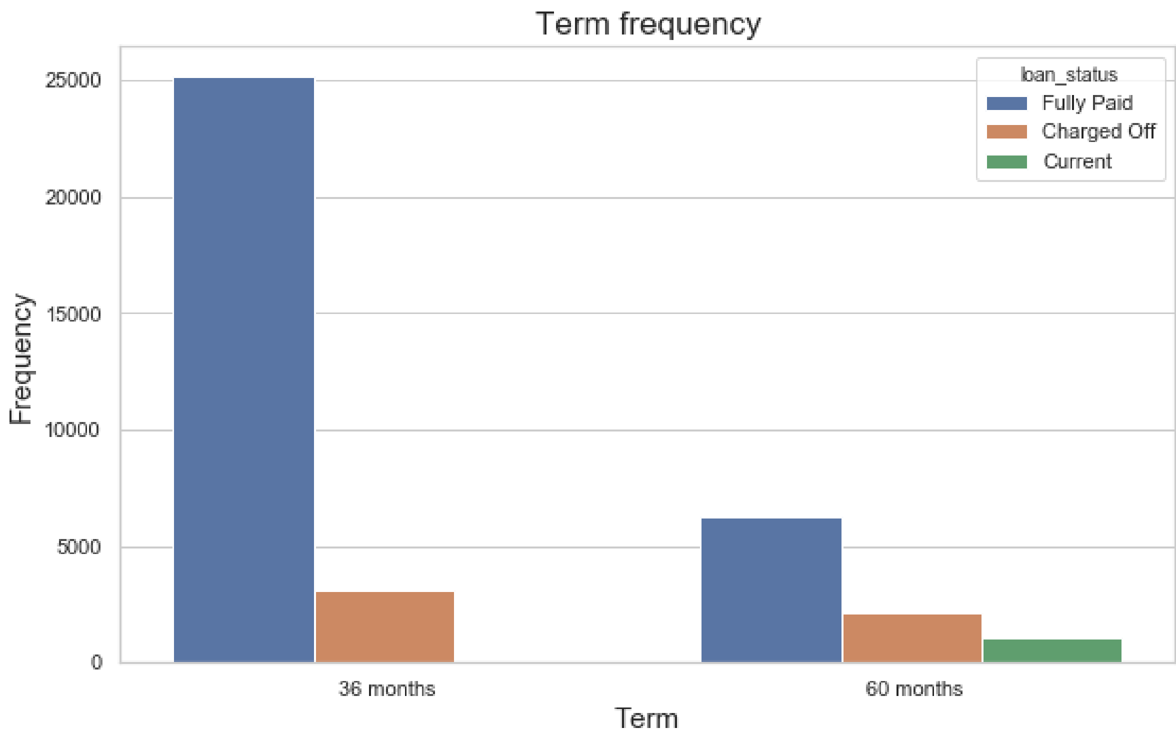
In [89]:

```
plt.figure(figsize=[10,6])
ax = sns.countplot(x = "term", hue="loan_status", data =lending_case)

ax.set_title("Term frequency", fontsize=17)
ax.set_ylabel("Frequency", fontsize=15)
ax.set_xlabel("Term", fontsize=15)
```

Out[89]:

```
Text(0.5, 0, 'Term')
```



Categorical variable issue_d analysis (summary)

There is not much significance as the year progresses, people tend to take more loan. It may be the possiblity that the economy is on the rise and consume expenses kept on increasing

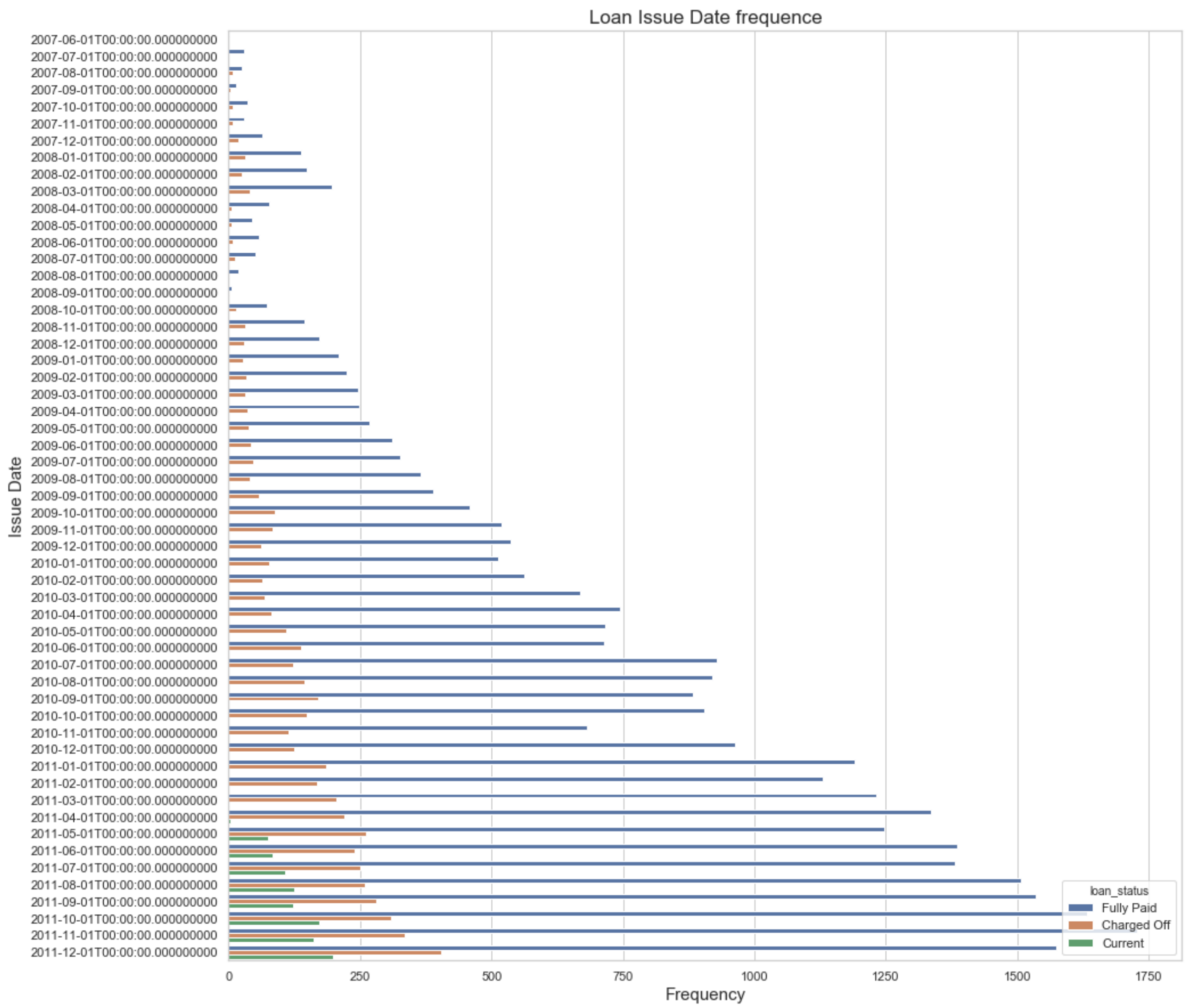
In [90]:

```
plt.figure(figsize=[15,15])
ax = sns.countplot(y = "issue_d", hue="loan_status", data =lending_case)

ax.set_title("Loan Issue Date frequence", fontsize=17)
ax.set_ylabel("Issue Date", fontsize=15)
ax.set_xlabel("Frequency", fontsize=15)
```

Out[90]:

```
Text(0.5, 0, 'Frequency')
```



Categorical variable (derived) issue_d_month analysis

Almost all the months contributes equally and december is the most, one of the reason could be people take higher loans in that month because income tax document submission starts in dec

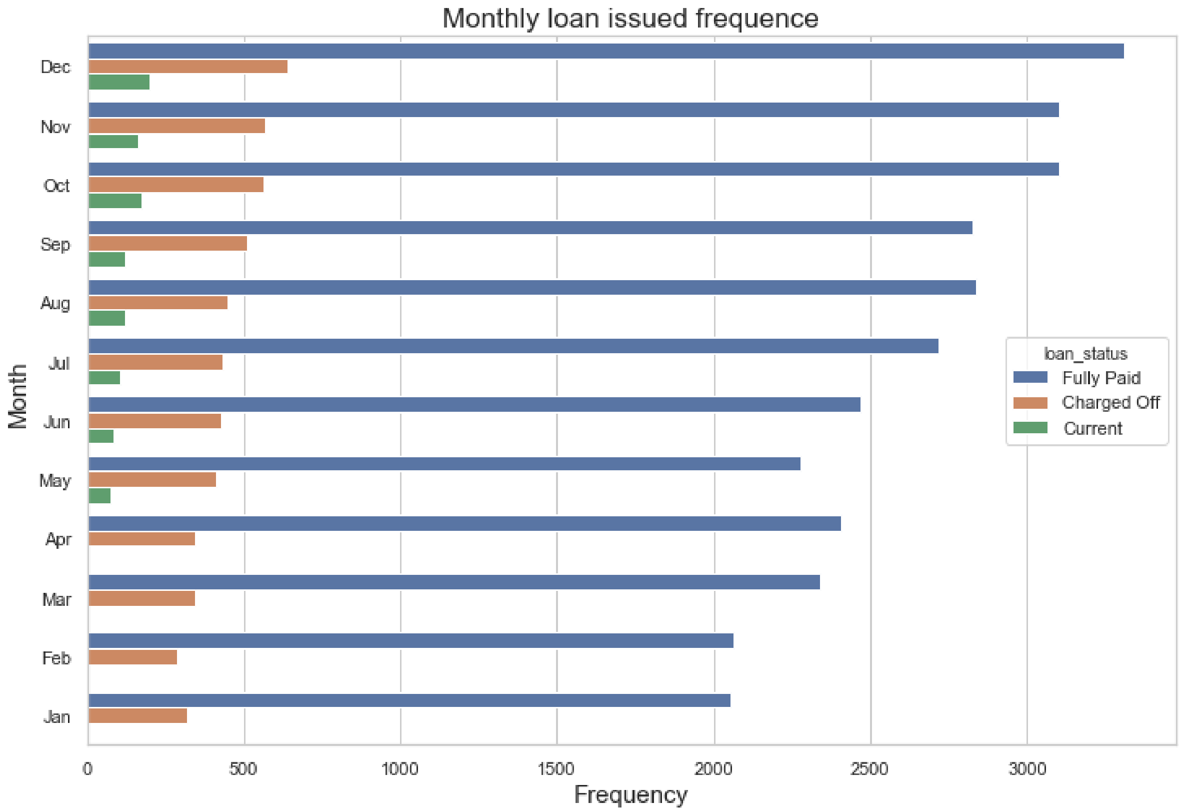
In [91]:

```
plt.figure(figsize=[12,8])
ax = sns.countplot(y = "issue_d_month", hue="loan_status", data =lending_case)

ax.set_title("Monthly loan issued frequence", fontsize=17)
ax.set_ylabel("Month", fontsize=15)
ax.set_xlabel("Frequency", fontsize=15)
```

Out[91]:

```
Text(0.5, 0, 'Frequency')
```



Find the relation between variables before proceeding further

In [92]:

```
lending_case.corr()
```

Out[92]:

	id	loan_amnt	int_rate	installment	emp_length	annual_inc	
id	1.000000	0.143636	0.068306	0.077759	0.094929	0.010192	0
loan_amnt	0.143636	1.000000	0.304068	0.958080	0.159974	0.275606	0
int_rate	0.068306	0.304068	1.000000	0.275539	0.014261	0.050449	0
installment	0.077759	0.958080	0.275539	1.000000	0.134669	0.282162	0
emp_length	0.094929	0.159974	0.014261	0.134669	1.000000	0.129679	0
annual_inc	0.010192	0.275606	0.050449	0.282162	0.129679	1.000000	-0
dti	0.097398	0.066441	0.107343	0.055008	0.052266	-0.123407	1
delinq_2yrs	-0.009397	-0.033849	0.156228	-0.021616	0.013264	0.022041	-0
inq_last_6mths	-0.041242	0.010460	0.136970	0.010963	0.009976	0.031708	0
open_acc	0.022126	0.177185	0.007814	0.175362	0.103972	0.163717	0
pub_rec	-0.020395	-0.053346	0.092906	-0.047868	0.050130	-0.018539	-0
revol_bal	0.012877	0.315477	0.097075	0.320237	0.157576	0.292629	0
total_acc	0.043063	0.249598	-0.049341	0.232635	0.206181	0.243002	0
out_prncp	0.176434	0.194125	0.134016	0.124162	0.050367	0.034371	0
out_prncp_inv	0.176389	0.193986	0.134199	0.124138	0.050301	0.034329	0
total_pymnt	0.132667	0.906503	0.301133	0.861729	0.146784	0.265782	0
total_pymnt_inv	0.212408	0.873721	0.296382	0.821851	0.154804	0.255952	0
total_rec_prncp	0.105434	0.874690	0.182320	0.856027	0.137816	0.268620	0
total_rec_int	0.166968	0.740349	0.522704	0.639293	0.129609	0.188775	0
total_rec_late_fee	-0.056380	0.046174	0.092162	0.056257	-0.015223	0.005507	-0
recoveries	0.033464	0.137571	0.121757	0.119838	0.025336	0.021098	0
collection_recovery_fee	-0.012862	0.075793	0.066648	0.076507	0.006536	0.015692	0
last_pymnt_amnt	0.117967	0.454970	0.153460	0.404841	0.081076	0.145313	0
policy_code	NaN	NaN	NaN	NaN	NaN	NaN	
acc_now_delinq	NaN	NaN	NaN	NaN	NaN	NaN	
issue_d_year	0.844713	0.119171	0.042128	0.044808	0.098566	0.010706	0

26 rows × 26 columns

In [93]:

```
#correlation matrix prints additional columsn where NAN matrix was determined
additional_colums_to_drop = ['policy_code', 'acc_now_delinq'] #['acc_now_delinq']
lending_case = lending_case.drop(columns = additional_colums_to_drop)
print("Shape after dropping columns with all missing values :: ", lending_case.shape
```

```
Shape after dropping columns with all missing values ::  (37868, 40)
```

Heat matrix between numerical variables

In [94]:

```
corr_matrix_column = ['loan_amnt', 'total_pymnt', 'total_pymnt_inv', 'int_rate', 'em
lending_case_corr = lending_case[corr_matrix_column]
lending_case_corr.corr()
```

Out[94]:

	loan_amnt	total_pymnt	total_pymnt_inv	int_rate	emp_length	annual_inc	
loan_amnt	1.000000	0.906503	0.873721	0.304068	0.159974	0.275606	0.0
total_pymnt	0.906503	1.000000	0.970702	0.301133	0.146784	0.265782	0.0
total_pymnt_inv	0.873721	0.970702	1.000000	0.296382	0.154804	0.255952	0.0
int_rate	0.304068	0.301133	0.296382	1.000000	0.014261	0.050449	0.1
emp_length	0.159974	0.146784	0.154804	0.014261	1.000000	0.129679	0.0
annual_inc	0.275606	0.265782	0.255952	0.050449	0.129679	1.000000	-0.1
dti	0.066441	0.065944	0.072825	0.107343	0.052266	-0.123407	1.0
issue_d_year	0.119171	0.119614	0.223810	0.042128	0.098566	0.010706	0.0

Observation from. heat map between numerical variable

positive correlationship

1. emp_length & loan_amnt : more experience employee can take high loan
2. loan_amt & annual_income: more loan is allowed for higher annual income
3. loan_amnt & int_rate : higher interest rate & loan_amnt (if person is defaulted, more loss for the firm)

negative correlationship

1. dti & annual_income : low debt ratio for higher annual income

In [95]:

```
loan_correlation = lending_case[corr_matrix_column].corr()
sns.clustermap(loan_correlation, annot=True, figsize=(12, 8), cmap = "Greens")
plt.show()
```

Bi-variate analysis, against charged off

```
1) annual_income
2) purpose
3) grade vs sub grade
4) interest rate
5) employee_length
6) address
7) verification
8) public bankruptcies
```

multiple variable analyis

Define the ranges for numerical variables for analysis

In [96]:

```
#annual income range
#lending_case['annual_inc'].describe()
bins = [0, 250000, 500000, 750000, 1000000, 1250000]
labels = ['0-250,000', '250,000-500,000', '500,000-750,000', '750,000-1000,000', '10
lending_case['annual_inc_range'] = pd.cut(lending_case['annual_inc'], bins, labels)
#lending_case['annual_inc_range'].unique()

#interest rate range
#lending_case['int_rate'].describe()
bins = [0, 5.0, 10.0, 15.0, 20.0, 25.0]
labels = ['0-5.0', '5.0-10.0', '10.0-15.0', '15.0-20.0', '20.0-25.0']
lending_case['int_rate_range'] = pd.cut(lending_case['int_rate'], bins, labels)
#lending_case['int_rate'].unique()


#loan amount range
#lending_case['loan_amnt_range'].describe()
bins = [0, 7000.0, 14000.0, 21000.0, 28000.0, 35000.0]
labels = ['0-7,000.0', '7,000.0-14,000.0', '14,000.0-21,000.0', '21,000.0-28,000.0',
lending_case['loan_amnt_range'] = pd.cut(lending_case['loan_amnt'], bins, labels)
#lending_case['loan_amnt_range'].unique()

#dti range
#lending_case['dti'].describe()
bins = [0, 10.0, 15.0, 20.0, 25.0, 30.0]
labels = ['0-10.0', '10.0-15.0', '15.0-20.0', '20.0-25.0', '25.0-30.0']
lending_case['dti_range'] = pd.cut(lending_case['dti'], bins, labels)
#lending_case['dti'].unique()
```

In [97]:

```
co_lending_case = lending_case[(lending_case['loan_status'] == 'Charged Off')]
co_lending_case = co_lending_case[(co_lending_case['annual_inc'] < 250000)]
```

Charged off vs annual_income

Bin the annual income in range with cut and plot the count plot graph

- Most of the defaulted loans are in annual income range of 0-250K i.e. people with higer income can easily pay the loan

In [98]:

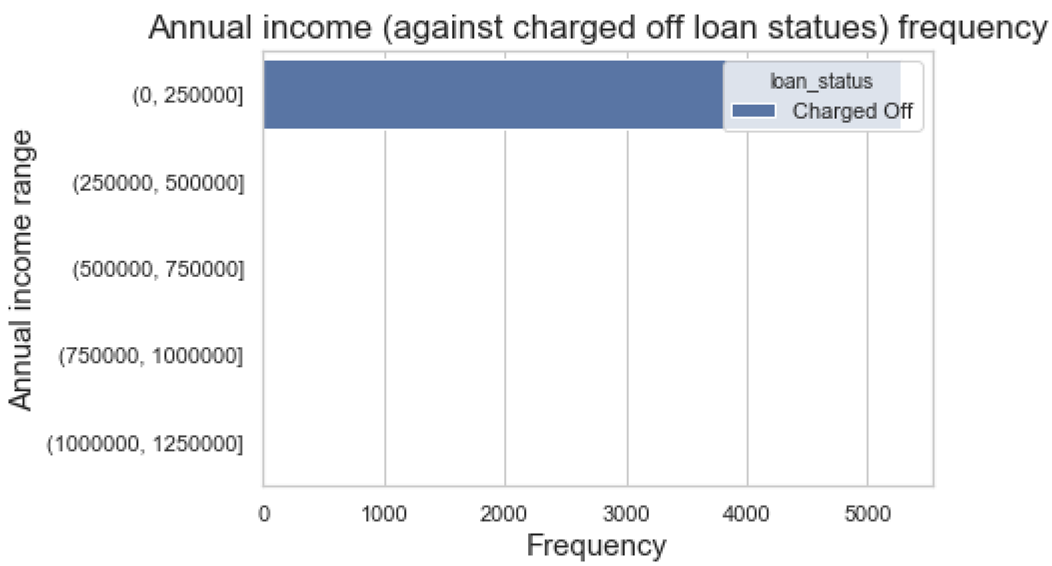
```
sns.set(style="whitegrid")
print(co_lending_case.groupby(['loan_status'])['annual_inc_range'].value_counts())
ax = sns.countplot(y = "annual_inc_range", hue="loan_status", data = co_lending_case

ax.set_title("Annual income (against charged off loan statues) frequency", fontsize=
ax.set_ylabel("Annual income range", fontsize=15)
ax.set_xlabel("Frequency", fontsize=15)
```

```
loan_status  annual_inc_range
Charged Off  (0, 250000]         5275
Name: annual_inc_range, dtype: int64
```

Out[98]:

```
Text(0.5, 0, 'Frequency')
```



Charged off vs purpose

Highest defaulted loan were in debt_consolidation, other, credit_card followed by small_business

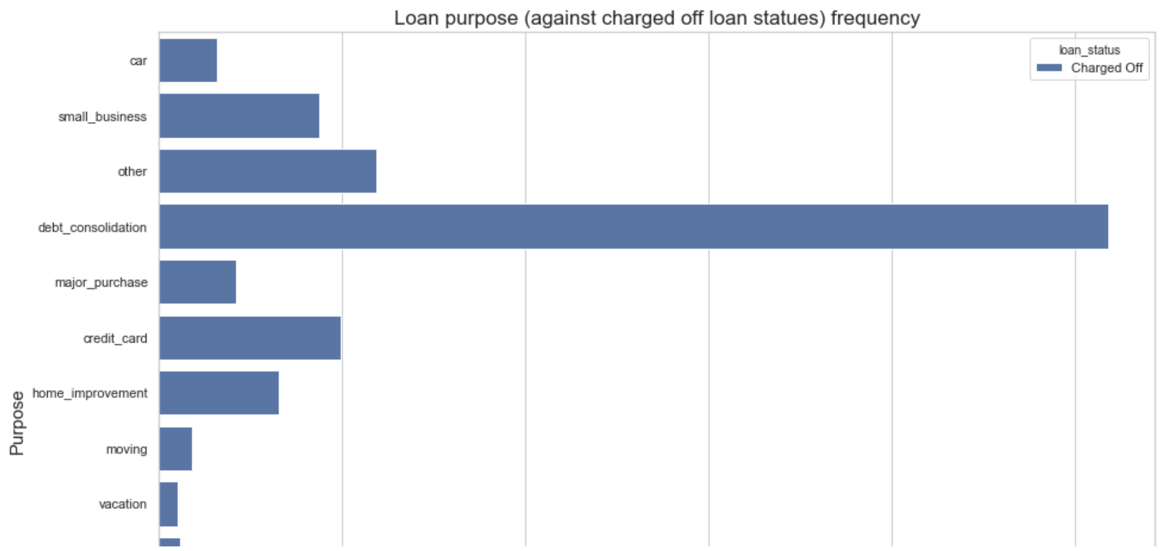
In [99]:

```
plt.figure(figsize=[15,12])
#print(co_lending_case.groupby(['loan_status'])['purpose'].value_counts())
ax = sns.countplot(y = "purpose", hue="loan_status", data = co_lending_case)

ax.set_title("Loan purpose (against charged off loan statues) frequency", fontsize=1
ax.set_xlabel("Frequency", fontsize=15)
ax.set_ylabel("Purpose", fontsize=15)
```

Out[99]:

```
Text(0, 0.5, 'Purpose')
```



Charged off vs Grade

Highest defaulted loan were in grades B,C, D

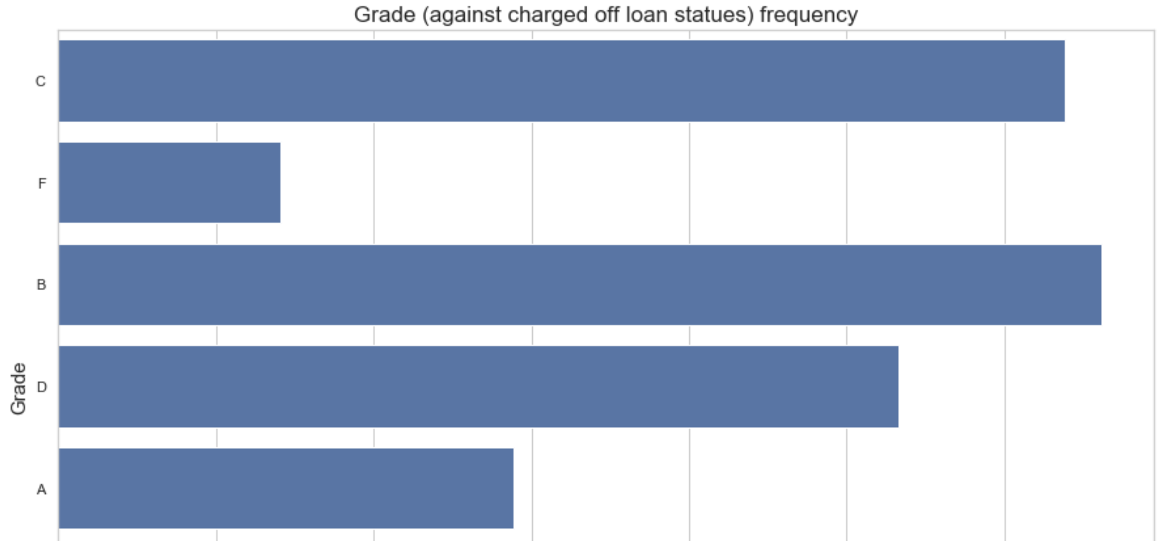
In [100]:

```
plt.figure(figsize=[15,10])
#print(co_lending_case.groupby(['loan_status'])['grade'].value_counts())
ax = sns.countplot(y = "grade", hue="loan_status", data = co_lending_case)

ax.set_title("Grade (against charged off loan statues) frequency", fontsize=17)
ax.set_xlabel("Frequency", fontsize=15)
ax.set_ylabel("Grade", fontsize=15)
```

Out[100]:

```
Text(0, 0.5, 'Grade')
```



Charged off vs Subgrade

loans in sub category of B & C are of defaulted most

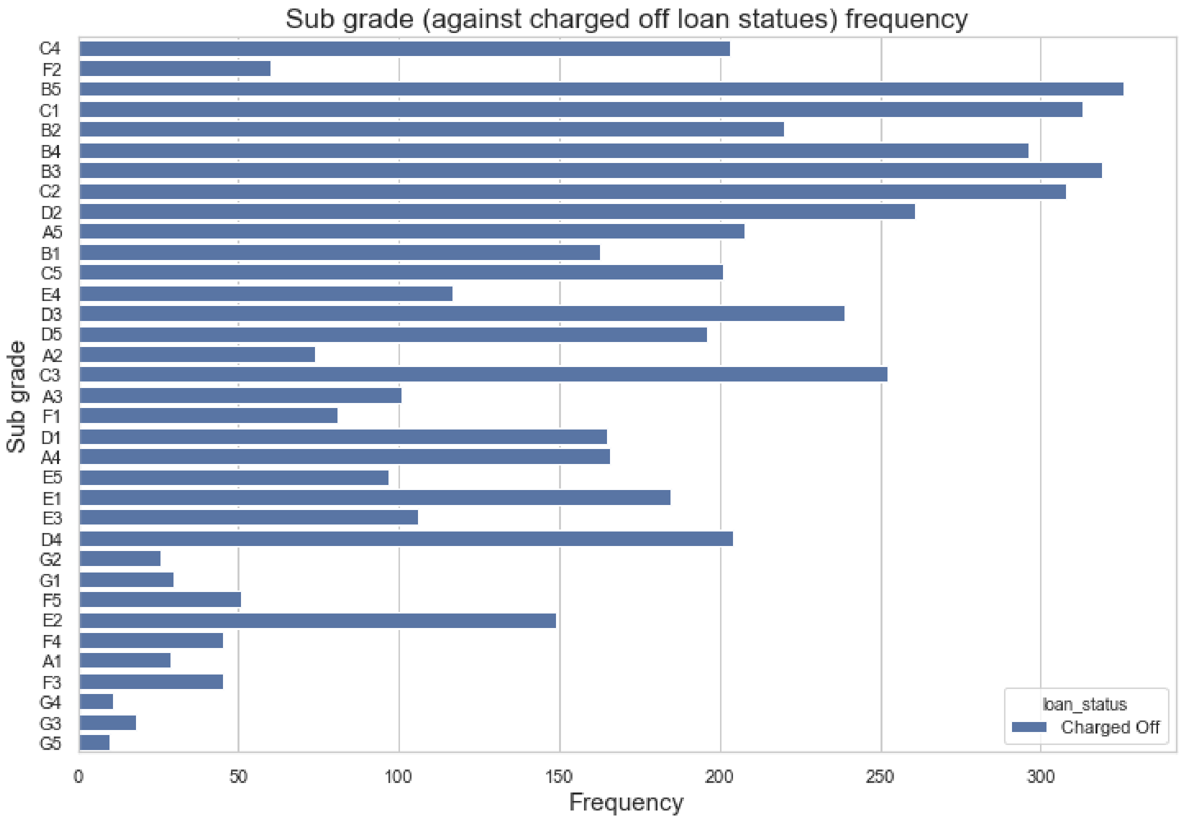
In [101]:

```
plt.figure(figsize=[12,8])
ax = sns.countplot(y = "sub_grade", hue="loan_status", data = co_lending_case)

ax.set_title("Sub grade (against charged off loan statues) frequency", fontsize=17)
ax.set_xlabel("Frequency", fontsize=15)
ax.set_ylabel("Sub grade", fontsize=15)
```

Out[101]:

```
Text(0, 0.5, 'Sub grade')
```



Charged off vs Interest rate

Highest interest rate for defaulted loan bar chart

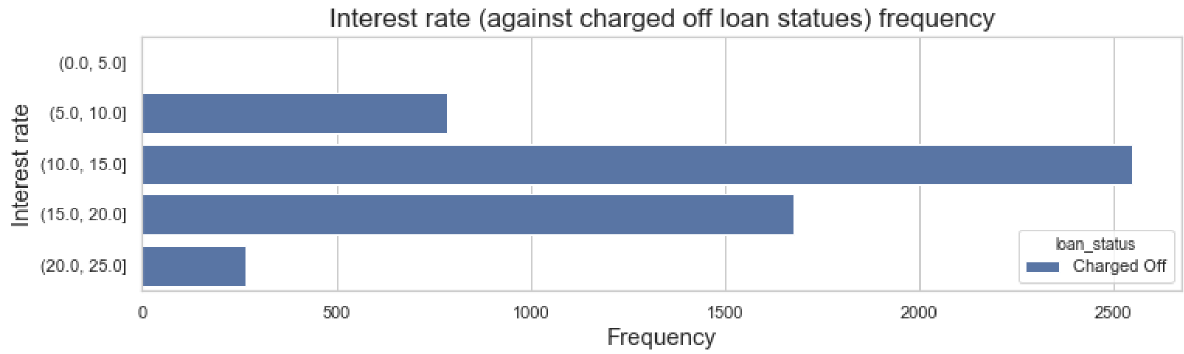
In [102]:

```
plt.figure(figsize=[12,3])
#print(co_lending_case.groupby(['loan_status'])['int_rate_range'].value_counts())
ax = sns.countplot(y = "int_rate_range", hue="loan_status", data = co_lending_case)

ax.set_title("Interest rate (against charged off loan statues) frequency", fontsize=
ax.set_xlabel("Frequency", fontsize=15)
ax.set_ylabel("Interest rate", fontsize=15)
```

Out[102]:

```
Text(0, 0.5, 'Interest rate')
```



Charged off vs employee length

Maximum defaulted loan were employee 10(+) years

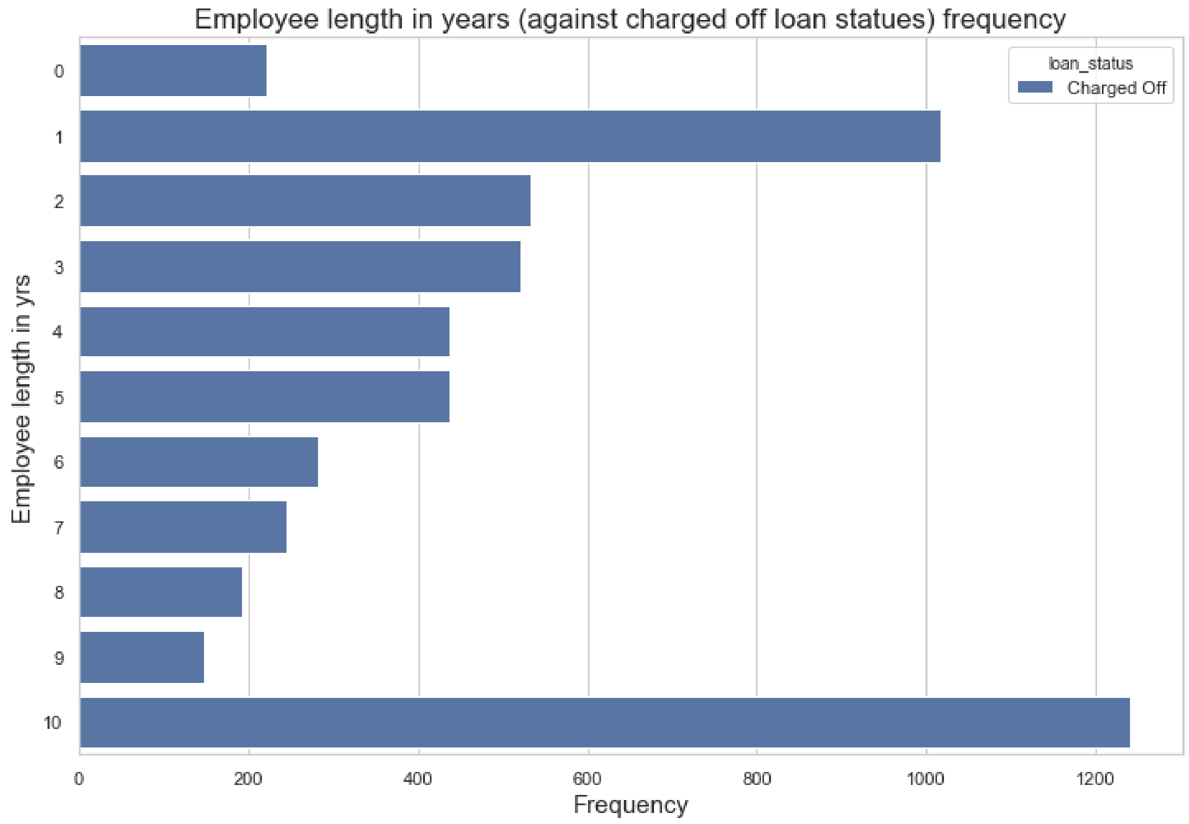
In [103]:

```
plt.figure(figsize=[12,8])
ax = sns.countplot(y = "emp_length", hue="loan_status", data = co_lending_case)

ax.set_title("Employee length in years (against charged off loan statues) frequency'
ax.set_xlabel("Frequency", fontsize=15)
ax.set_ylabel("Employee length in yrs", fontsize=15)
```

Out[103]:

```
Text(0, 0.5, 'Employee length in yrs')
```



Charged off vs Address (addr_state)

Maximum defaulted loan were for state CA followed by FL, NY

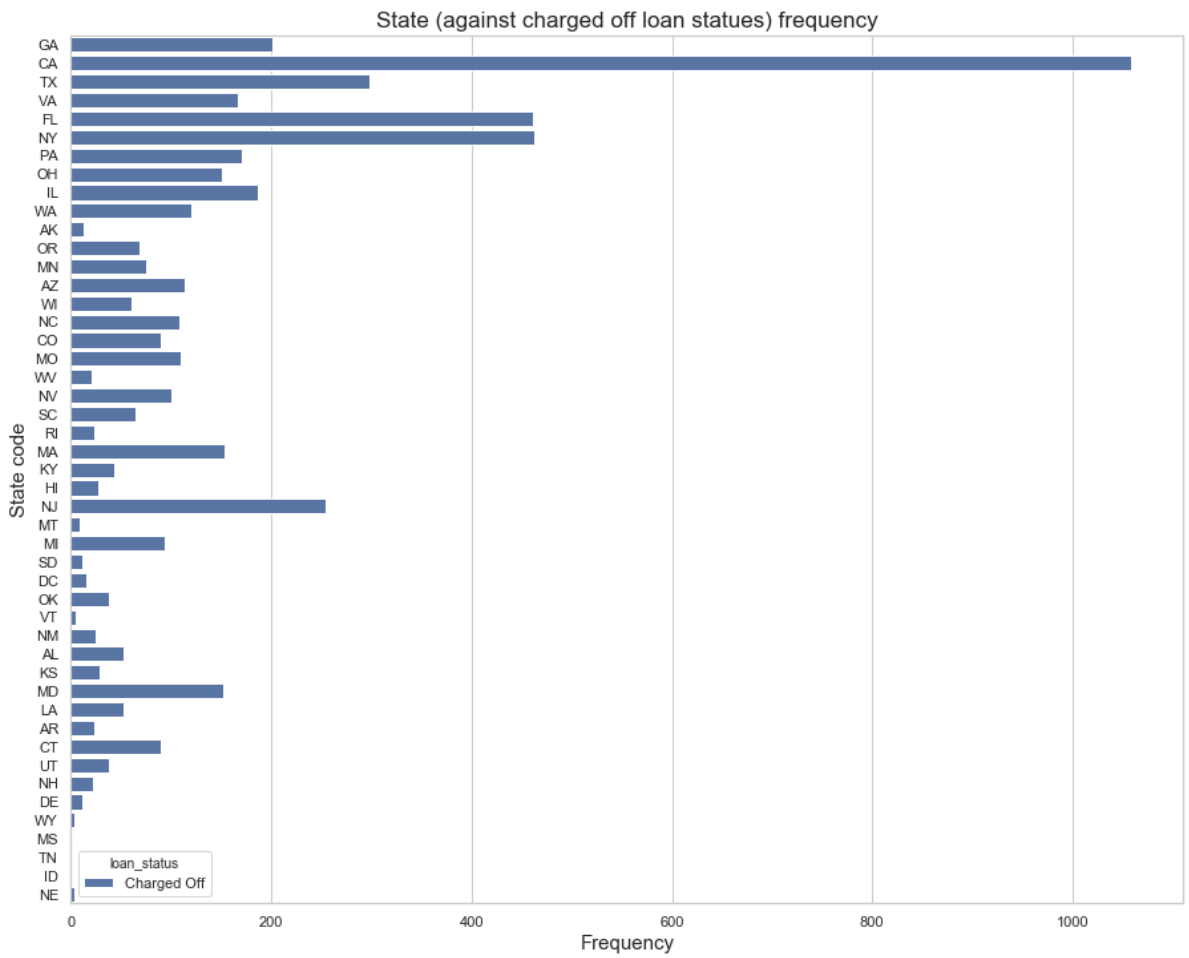
In [104]:

```
plt.figure(figsize=[15,12])
ax = sns.countplot(y = "addr_state", hue="loan_status", data = co_lending_case)

ax.set_title("State (against charged off loan statues) frequency", fontsize=17)
ax.set_xlabel("Frequency", fontsize=15)
ax.set_ylabel("State code", fontsize=15)
```

Out[104]:

```
Text(0, 0.5, 'State code')
```



Charged off vs Verification

Maximum defaulted loan were for status Not Verified

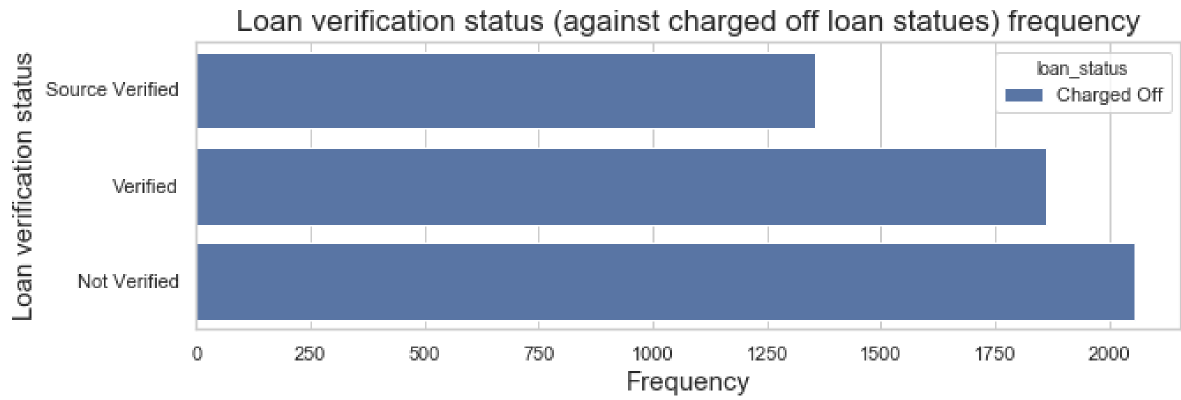
In [105]:

```
plt.figure(figsize=[10,3])
#print(co_lending_case.groupby(['loan_status'])['int_rate_range'].value_counts())
ax = sns.countplot(y = "verification_status", hue="loan_status", data = co_lending_c

ax.set_title("Loan verification status (against charged off loan statues) frequency'
ax.set_xlabel("Frequency", fontsize=15)
ax.set_ylabel("Loan verification status", fontsize=15)
```

Out[105]:

```
Text(0, 0.5, 'Loan verification status')
```



Charged off vs public bankruptcies

Maximum defaulted loan were for public recorded bankruptcies with zero

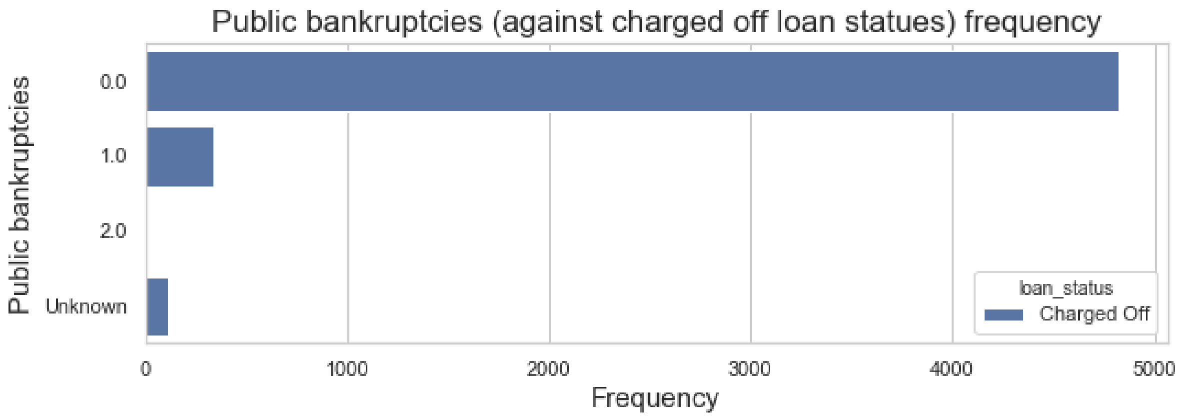
In [106]:

```
plt.figure(figsize=[10,3])
#print(co_lending_case.groupby(['loan_status'])['pub_rec_bankruptcies'].value_counts
ax = sns.countplot(y = "pub_rec_bankruptcies", hue="loan_status", data = co_lending_

ax.set_title("Public bankruptcies (against charged off loan statues) frequency", for
ax.set_xlabel("Frequency", fontsize=15)
ax.set_ylabel("Public bankruptcies", fontsize=15)
```

Out[106]:

```
Text(0, 0.5, 'Public bankruptcies')
```



Bi-variate analyis further two attributes at a time

1. Loan amount vs purpose of loan
2. Interest rate vs Term of loan
3. Grade vs Interest Rate
4. Year vs interest rate
5. Loan amount vs Interest rate
6. Dti vs interest rate
7. Annual income across grade

Loan amount vs purpose of loan

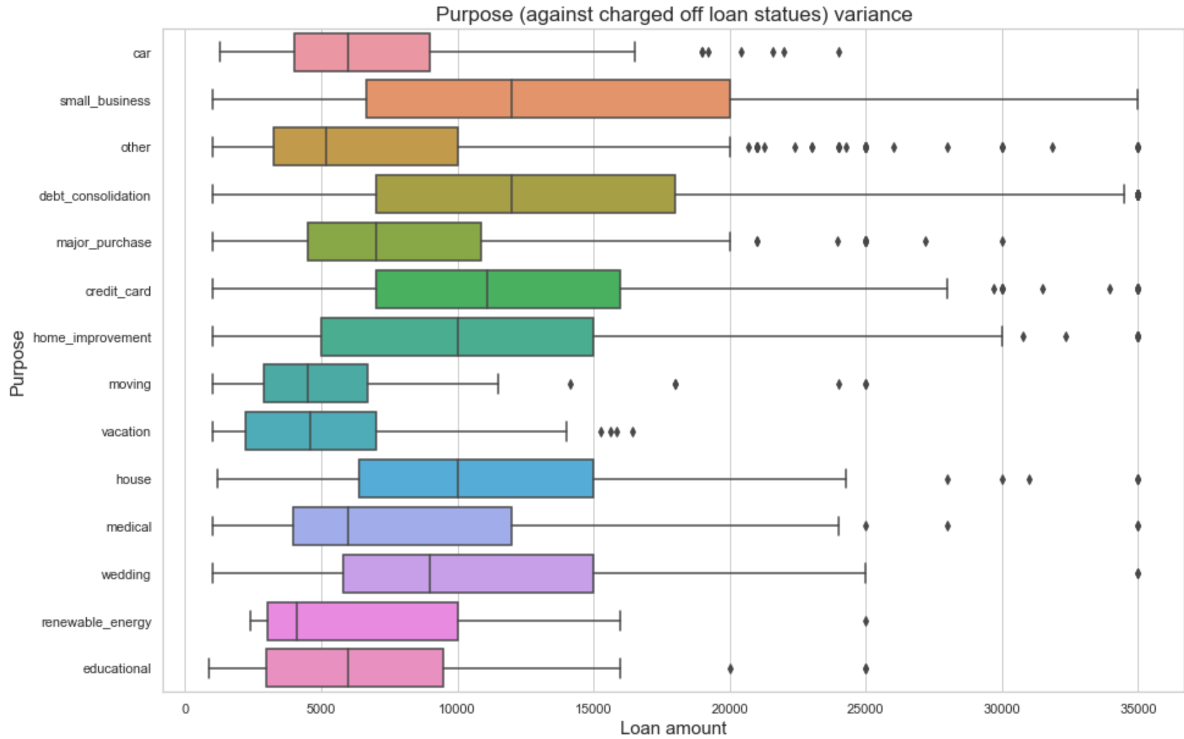
highest loan amount in (90-95 percentile) was taken for small_business followed by debit_consolidation and credit_case median (majority of 50%) was loan taken in debt_consolidation followed by credit_card, small_business, house

In [107]:

```
#lending_case[['loan_amnt', 'purpose']]

plt.figure(figsize=[15,10])
ax = sns.boxplot(co_lending_case['loan_amnt'], co_lending_case['purpose'])

ax.set_title("Purpose (against charged off loan statues) variance", fontsize=17)
ax.set_ylabel("Purpose", fontsize=15)
ax.set_xlabel("Loan amount", fontsize=15)
plt.show()
```



Interest rate vs Term of loan

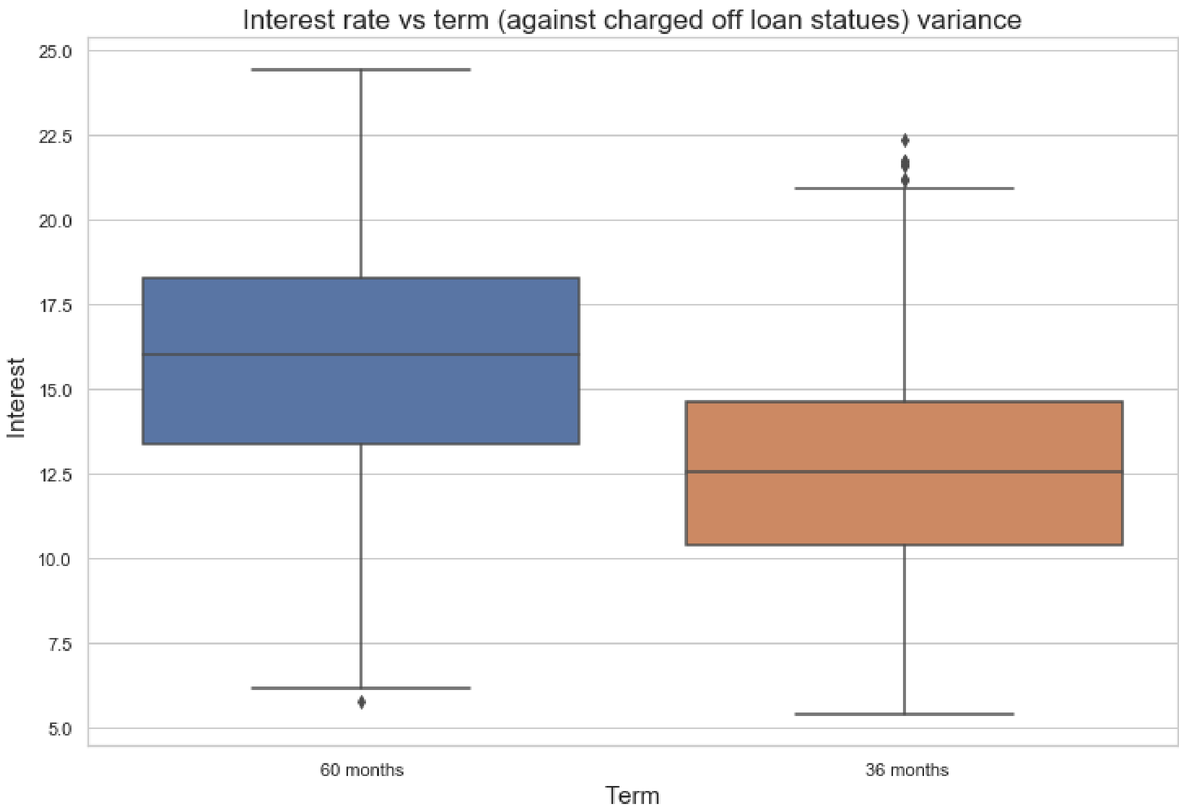
Loan taken for 60 months has higher interest rate as chances of defaulting may be higher

In [108]:

```
#lending_case[['int_rate', 'term']]
plt.figure(figsize=[12,8])
ax = sns.boxplot(co_lending_case['term'], lending_case['int_rate'])

ax.set_title("Interest rate vs term (against charged off loan statues) variance", fc
ax.set_xlabel("Term", fontsize=15)
ax.set_ylabel("Interest ", fontsize=15)

plt.show()
```



Grade vs Interest Rate

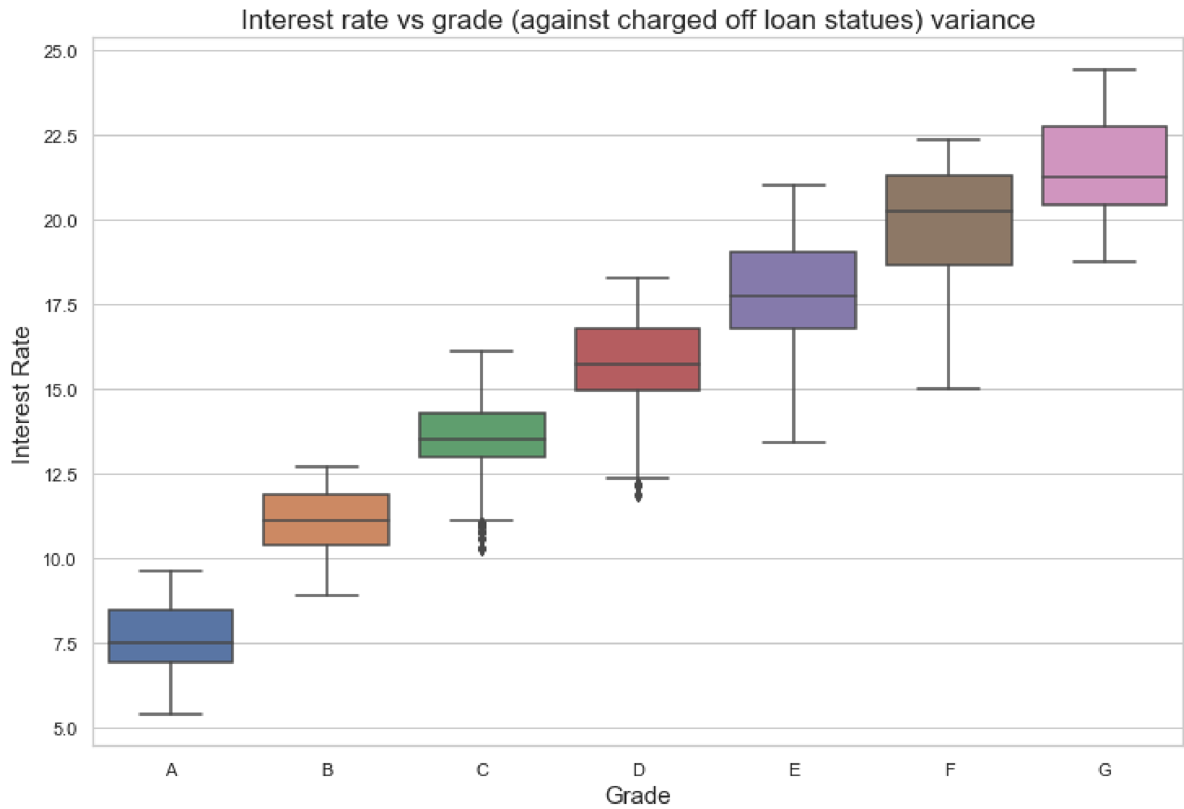
Loans with bad grade has high interest rate; as the chances of defaulting loans are higher

In [109]:

```
plt.figure(figsize=[12,8])
ax = sns.boxplot(co_lending_case['grade'].sort_values(ascending=True), lending_case[

ax.set_title("Interest rate vs grade (against charged off loan statues) variance", f
ax.set_ylabel("Interest Rate", fontsize=15)
ax.set_xlabel("Grade", fontsize=15)

plt.show()
```



Year vs interest rate

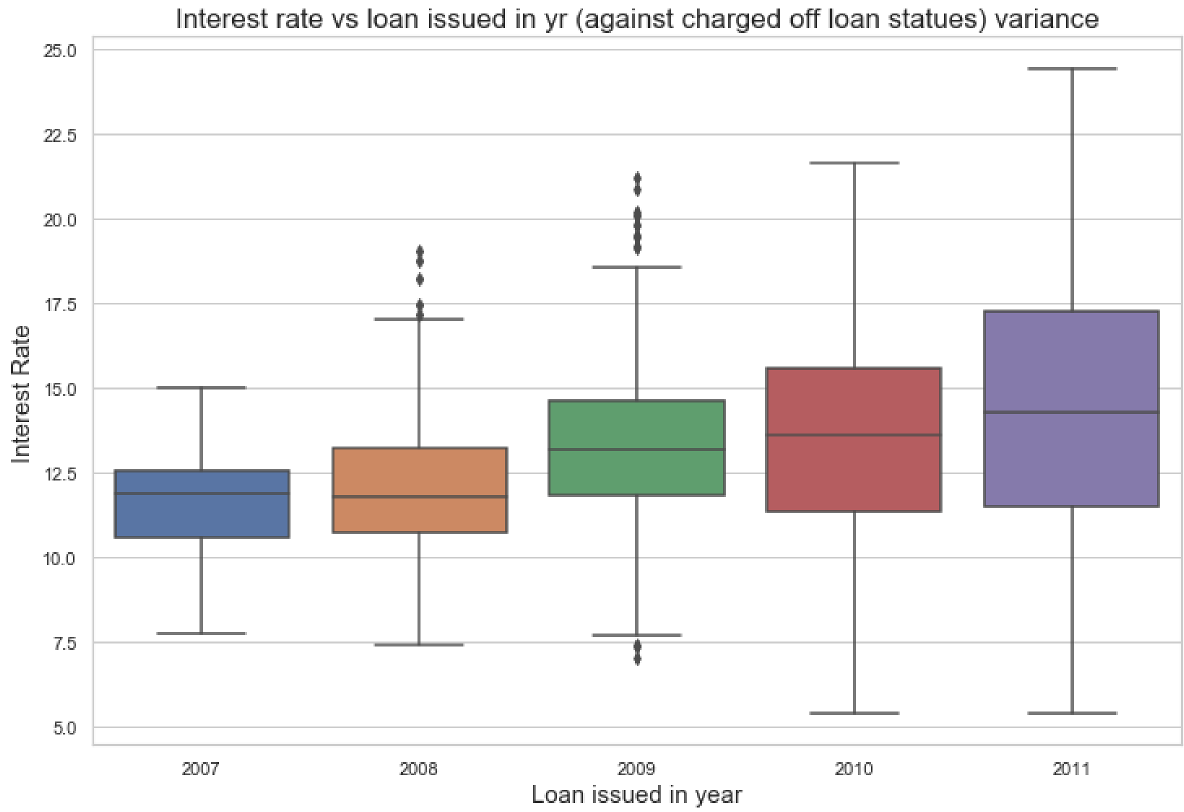
- As the year progresses interest rate bars keep increasing. possibility economoy is on rise.
- 25% percentile was almost at the same level as the people income kept increasing

In [110]:

```
plt.figure(figsize=[12,8])
ax = sns.boxplot(co_lending_case['issue_d_year'].sort_values(ascending=True), lendir

ax.set_title("Interest rate vs loan issued in yr (against charged off loan statues)
ax.set_ylabel("Interest Rate", fontsize=15)
ax.set_xlabel("Loan issued in year", fontsize=15)

plt.show()
```



Loan amount vs interest rate

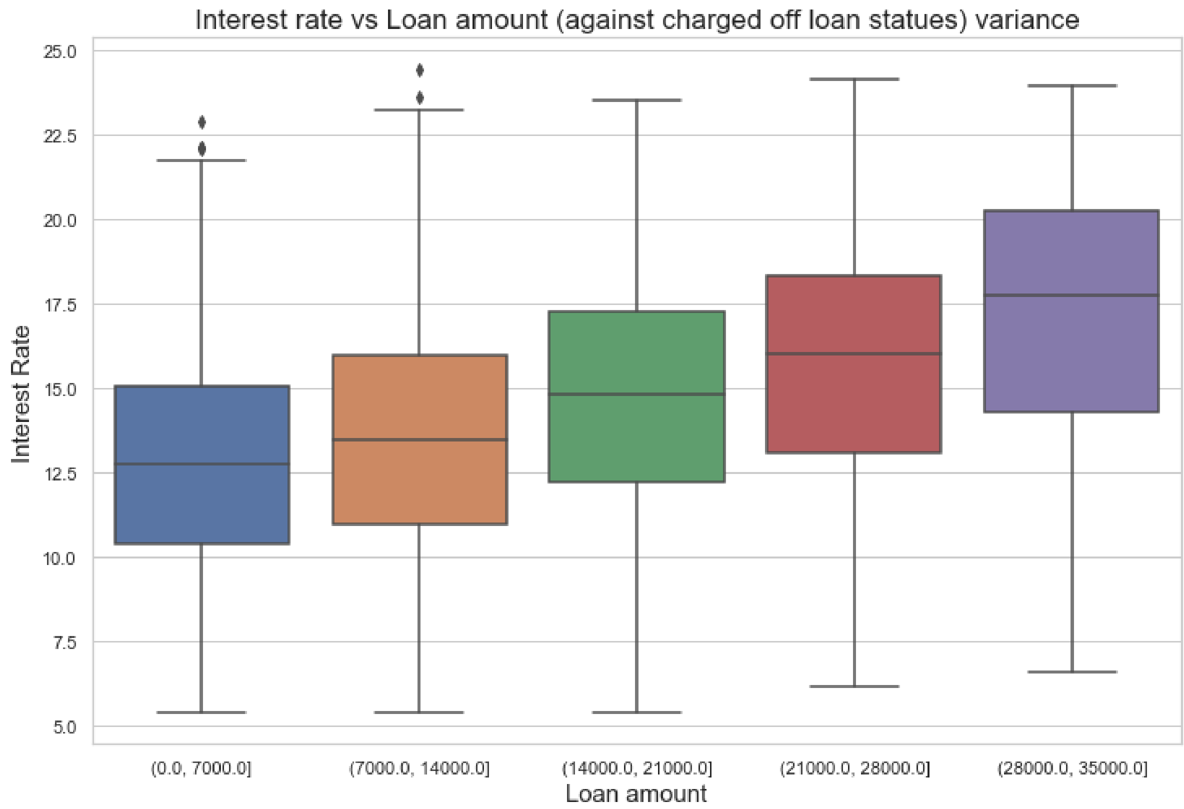
- As the loan amount increases, interest rate also increases
- 25%, 50%, 75% also increased the loan amount increased

In [111]:

```
plt.figure(figsize=[12,8])
ax = sns.boxplot(co_lending_case['loan_amnt_range'], co_lending_case['int_rate'])

ax.set_title("Interest rate vs Loan amount (against charged off loan statues) variar
ax.set_ylabel("Interest Rate", fontsize=15)
ax.set_xlabel("Loan amount", fontsize=15)

plt.show()
```



Dti vs interest rate

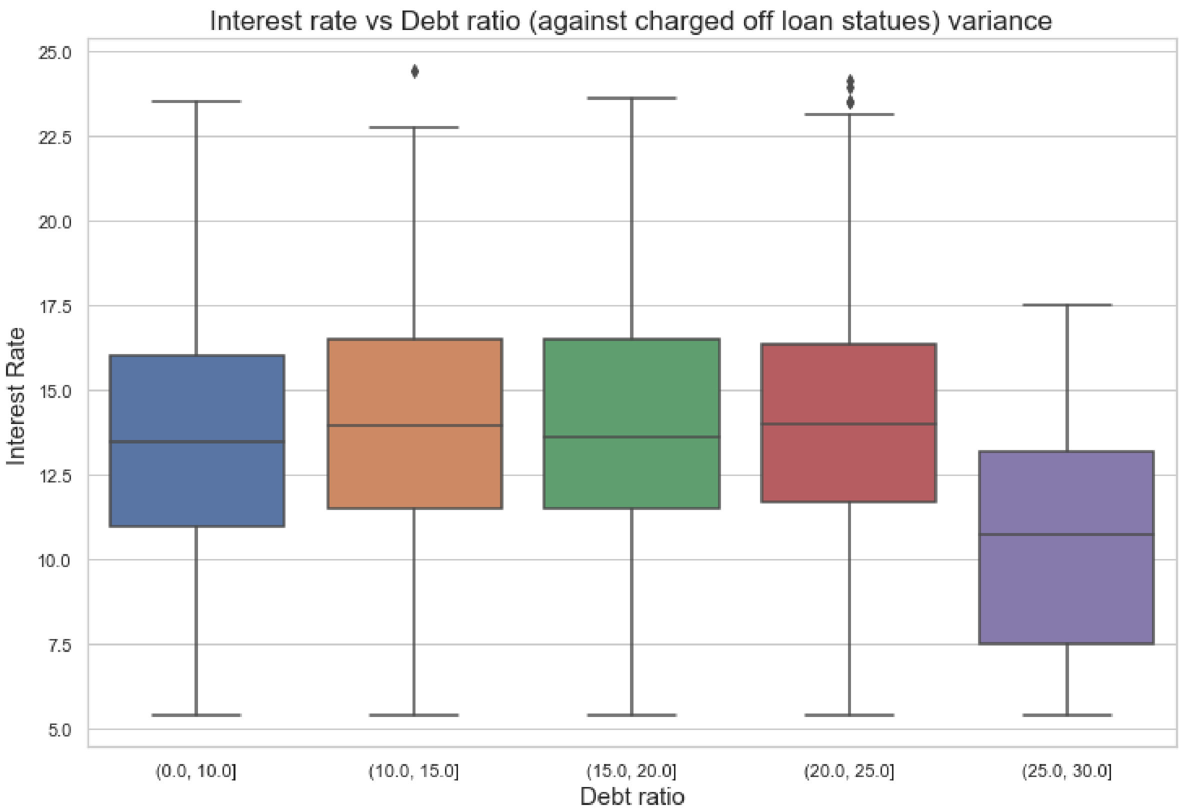
- Variation remains same across all the dti_ranges
- For range 25.0-30.0, it may be due to outlier and people might have exceptionally high income

In [112]:

```
plt.figure(figsize=[12,8])
ax = sns.boxplot(co_lending_case['dti_range'], co_lending_case['int_rate'])

ax.set_title("Interest rate vs Debt ratio (against charged off loan statues) varianc
ax.set_ylabel("Interest Rate", fontsize=15)
ax.set_xlabel("Debt ratio", fontsize=15)

plt.show()
```



Annual income vs grade

- Mostly people from income range < 250000 were defaulted
- From Income range analysis its clear people from <= 250000 are defaulted

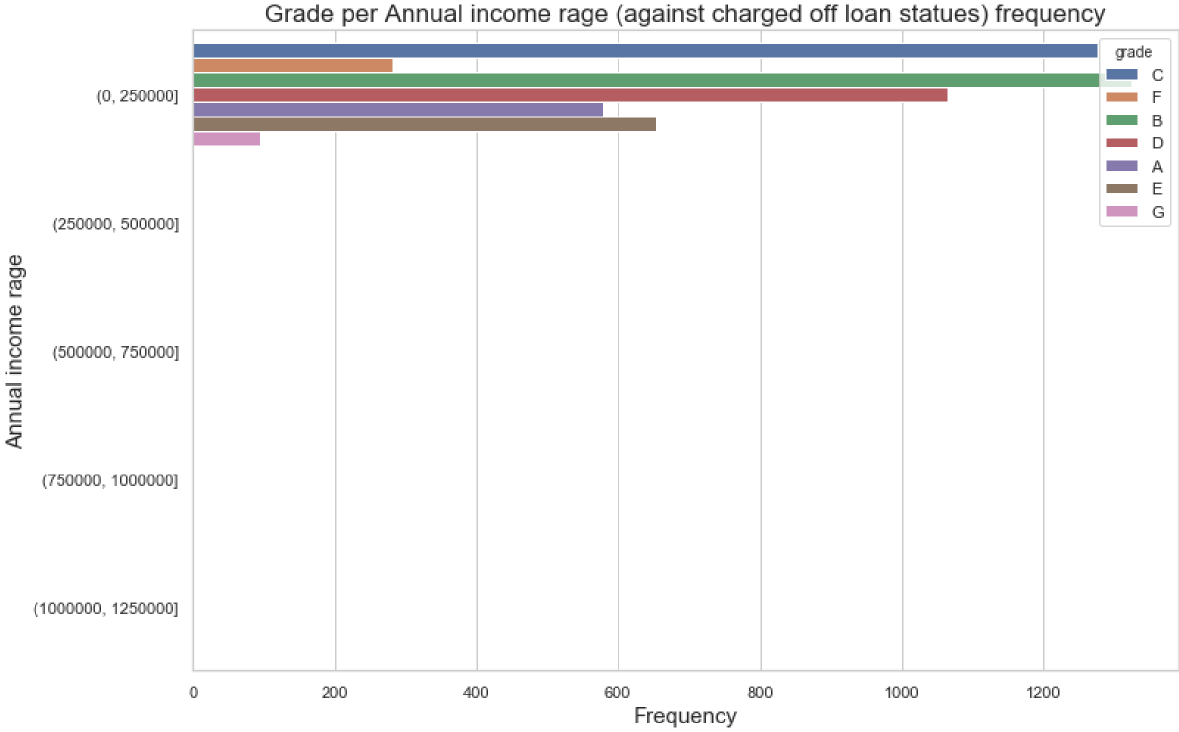
In [113]:

```
plt.figure(figsize=[12,8])
ax = sns.countplot(y = "annual_inc_range", hue="grade", data = co_lending_case)

ax.set_title("Grade per Annual income rage (against charged off loan statues) freque
ax.set_ylabel("Annual income rage", fontsize=15)
ax.set_xlabel("Frequency", fontsize=15)
```

Out[113]:

```
Text(0.5, 0, 'Frequency')
```



Multivariate analysis

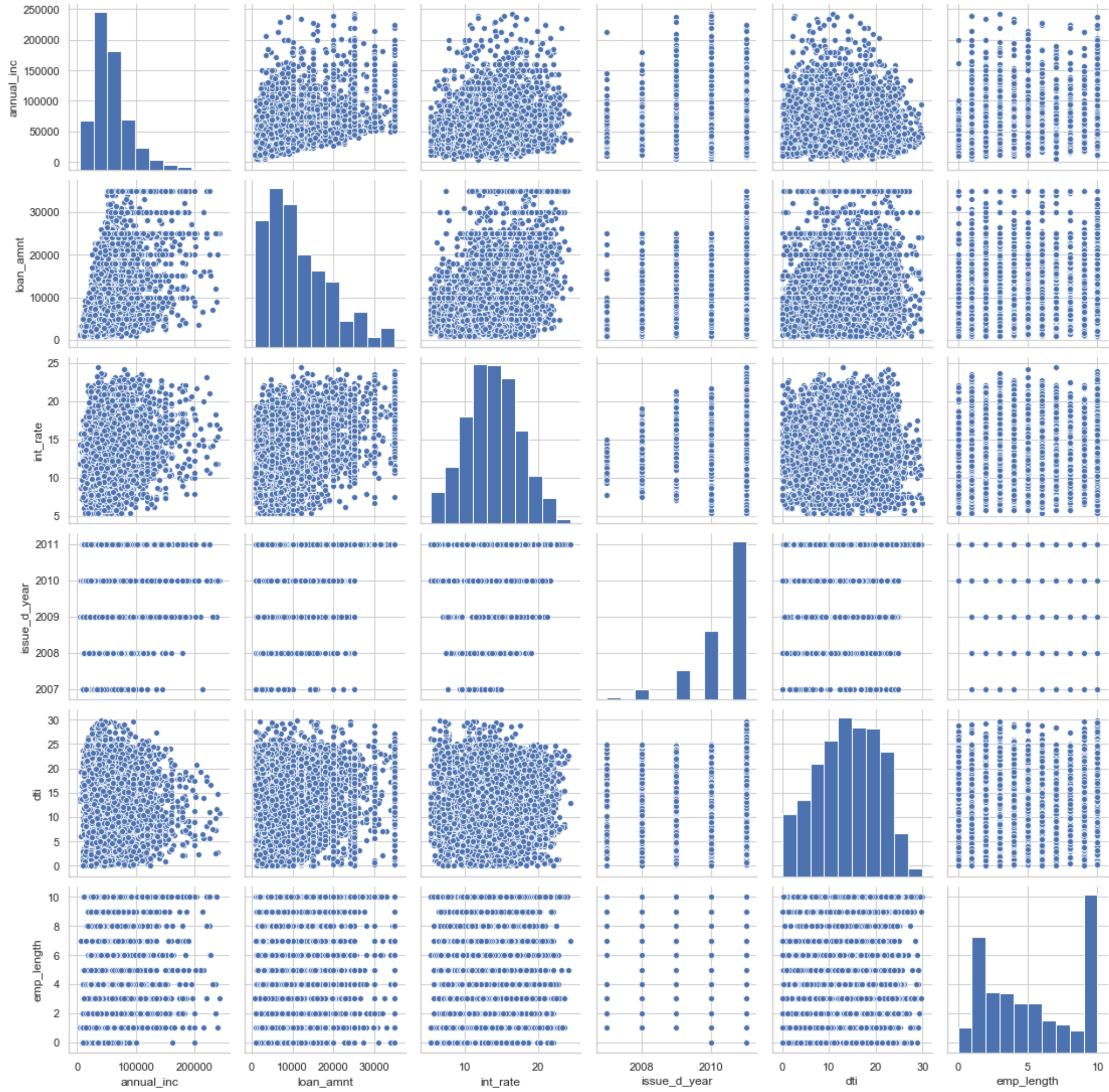
Pair plot between different numerical variables (Summary for charged off loans)

- Interest rate fairly distributed across annual income
- Annual income kept increasting as the year progresses
- Interest rates kept increasting as the year progresses possibly economy was on higher side
- dti debit ratio fairly distributed across annual income < 200K& interest rate

In [114]:

```
plt.figure(figsize=[12,8])
sns.pairplot(co_lending_case[['annual_inc', 'loan_amnt', 'int_rate', 'issue_d_year',
plt.show()
```

```
<Figure size 864x576 with 0 Axes>
```



In [ ]:

In [ ]: