DATA 6320 - Scenario 3

Program Sections

Contents

- **Business Understanding**
  - Available resources, problems, goals
- **Data Understanding**
  - What data do you have available to you?
  - Install or Load tools or applications
    - Programming – Jupyter notebooks for Python or R Studio for R
    - BI/spreadsheets – Excel – PowerPivot - Tableau
  - Import or download the data
  - Format the data
  - View, explore, and summarize the data
- **Data Preparation**
  - Remove Columns
  - Remove Rows
  - Fill in null values
  - Replace or remove mistakes
  - **Remove outliers**
  - **Recode categorical or numerical features**
  - Construct new data feature engineering
  - For Supervised Learning, create X and Y
- **Modeling**
  - Split the data (Train/Test Split)
  - Transform the data
  - Setup models for machine learning/AI processes
  - Can also include developing the outline for visuals, dashboards or reports
- **Evaluation**
  - Hyper-parameter tuning
- **Deployment of models**

BUSINESS UNDERSTANDING

Business Objective

- What is the relationship between annual income and loan amount. Is it a good predictor?
- Can we predict the amount for a loan?
- What features are most important in predicting loan amount?

Technical Objective

- Review different data cleansing techniques to continue to improve
- Conduct a simple regression using scikit-learn
- Conduct a multiple regression using statsmodels
- Learn and perform Lasso and Ridge regression and tune its parameters

---

DATA UNDERSTANDING

Import Libraries

In [1]:

```
#Code Block 01
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

pd.set_option('display.max_columns',500)

plt.style.use('seaborn-v0_8-colorblind') #a style that can be used for plots
```

```
sns.set_style('whitegrid')
```

In [2]:

```
#obtain the list of available styles

plt.style.available
```

Out[2]:

```
['Solarize_Light2',
 '_classic_test_patch',
 '_mpl-gallery',
 '_mpl-gallery-nogrid',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn-v0_8',
 'seaborn-v0_8-bright',
 'seaborn-v0_8-colorblind',
 'seaborn-v0_8-dark',
 'seaborn-v0_8-dark-palette',
 'seaborn-v0_8-darkgrid',
 'seaborn-v0_8-deep',
 'seaborn-v0_8-muted',
 'seaborn-v0_8-notebook',
 'seaborn-v0_8-paper',
 'seaborn-v0_8-pastel',
 'seaborn-v0_8-poster',
 'seaborn-v0_8-talk',
 'seaborn-v0_8-ticks',
 'seaborn-v0_8-white',
 'seaborn-v0_8-whitegrid',
 'tableau-colorblind10']
```

Import the data and create dataframes that can then be cleansed, recoded, transformed, and split.

Import Data

Top

Look for outliers

In [3]:

```
#Code Block 02

df_loandata_clean = pd.read_csv('data/DATA6320_Scenario3.csv', index_col = 0, header = 0
df_loandata_clean.info()
```

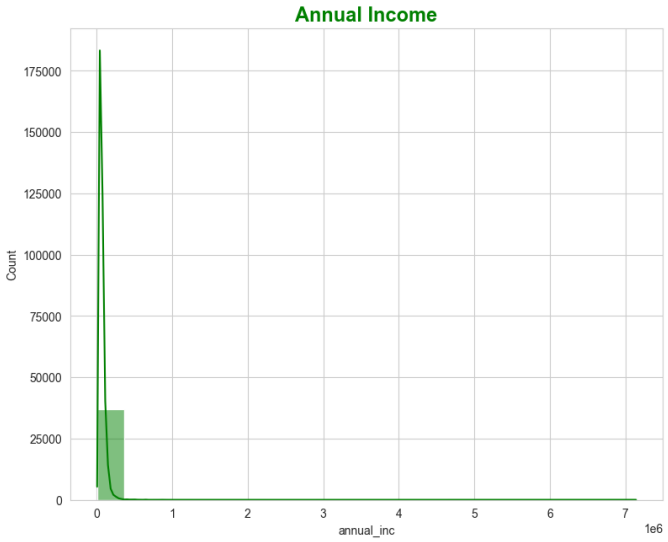
```
<class 'pandas.core.frame.DataFrame'>
Index: 37132 entries, 0 to 37131
Data columns (total 27 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   member_id                    37132 non-null  int64
 1   loan_amnt                    37132 non-null  int64
 2   term                         37132 non-null  int64
 3   int_rate                     37132 non-null  float64
 4   annual_inc                   37132 non-null  float64
 5   delinq_2yrs                  37132 non-null  int64
 6   inq_last_6mths               37132 non-null  int64
 7   mths_since_last_delinq       37132 non-null  int64
 8   mths_since_last_record       37132 non-null  int64
 9   open_acc                     37132 non-null  int64
 10  pub_rec                      37132 non-null  int64
 11  revol_bal                    37132 non-null  int64
 12  total_acc                    37132 non-null  int64
 13  total_debt_paid              37132 non-null  float64
 14  princ_int_ratio              37132 non-null  float64
 15  collections_12_mths_ex_med   37132 non-null  int64
 16  mths_since_last_major_derog  37132 non-null  int64
 17  acc_now_delinq               37132 non-null  int64
 18  tot_coll_amt                 37132 non-null  int64
 19  tot_cur_bal                  37132 non-null  int64
 20  total_credit_rv              37132 non-null  int64
 21  revol_util                   37132 non-null  float64
 22  sub_grade                    37132 non-null  object
 23  emp_length                   37132 non-null  int64
 24  home_ownership               37132 non-null  object
 25  loan_status                  37132 non-null  object
 26  reason                       37132 non-null  object
dtypes: float64(5), int64(18), object(4)
memory usage: 7.9+ MB
```

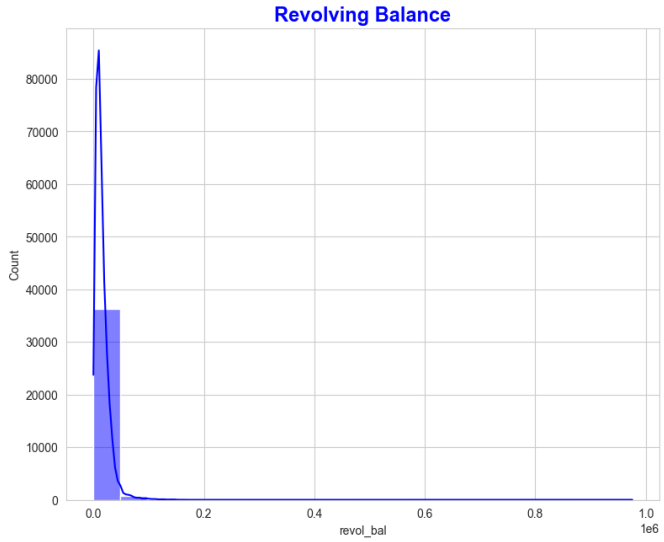
In [4]:

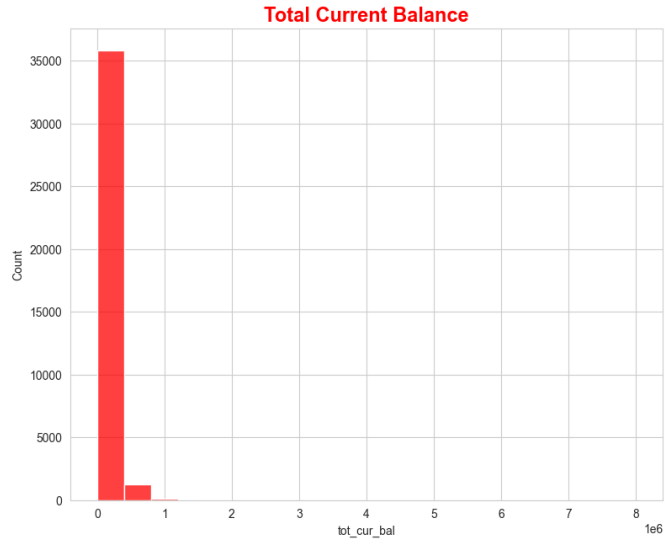
```
#Code Block 03

plt.figure(figsize=(20,16))

plt.subplot(221)
plt.title('Annual Income', fontweight='bold', color = 'green', fontsize='17', horizontal
sns.histplot(df_loandata_clean['annual_inc'], color="g", bins = 20, kde=True)

plt.subplot(222)
plt.title('Revolving Balance', fontweight='bold', color = 'blue', fontsize='17', horizon
sns.histplot(df_loandata_clean['revol_bal'], color="b", bins = 20, kde=True)

plt.subplot(223)
plt.title('Total Current Balance', fontweight='bold', color = 'red', fontsize='17', hori
sns.histplot(df_loandata_clean['tot_cur_bal'], color="r",  bins = 20, kde=False)

plt.subplot(224)
plt.title('Employee Length', fontweight='bold', color = 'black', fontsize='17', horizont
sns.histplot(df_loandata_clean['emp_length'], color="y",  bins = 20, kde=True )
```

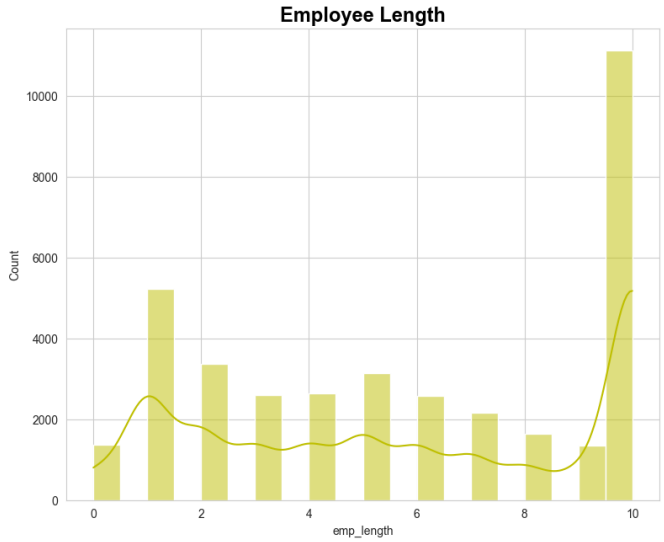
Out[4]:

```
<Axes: title={'center': 'Employee Length'}, xlabel='emp_length', ylabel='Count'>
```







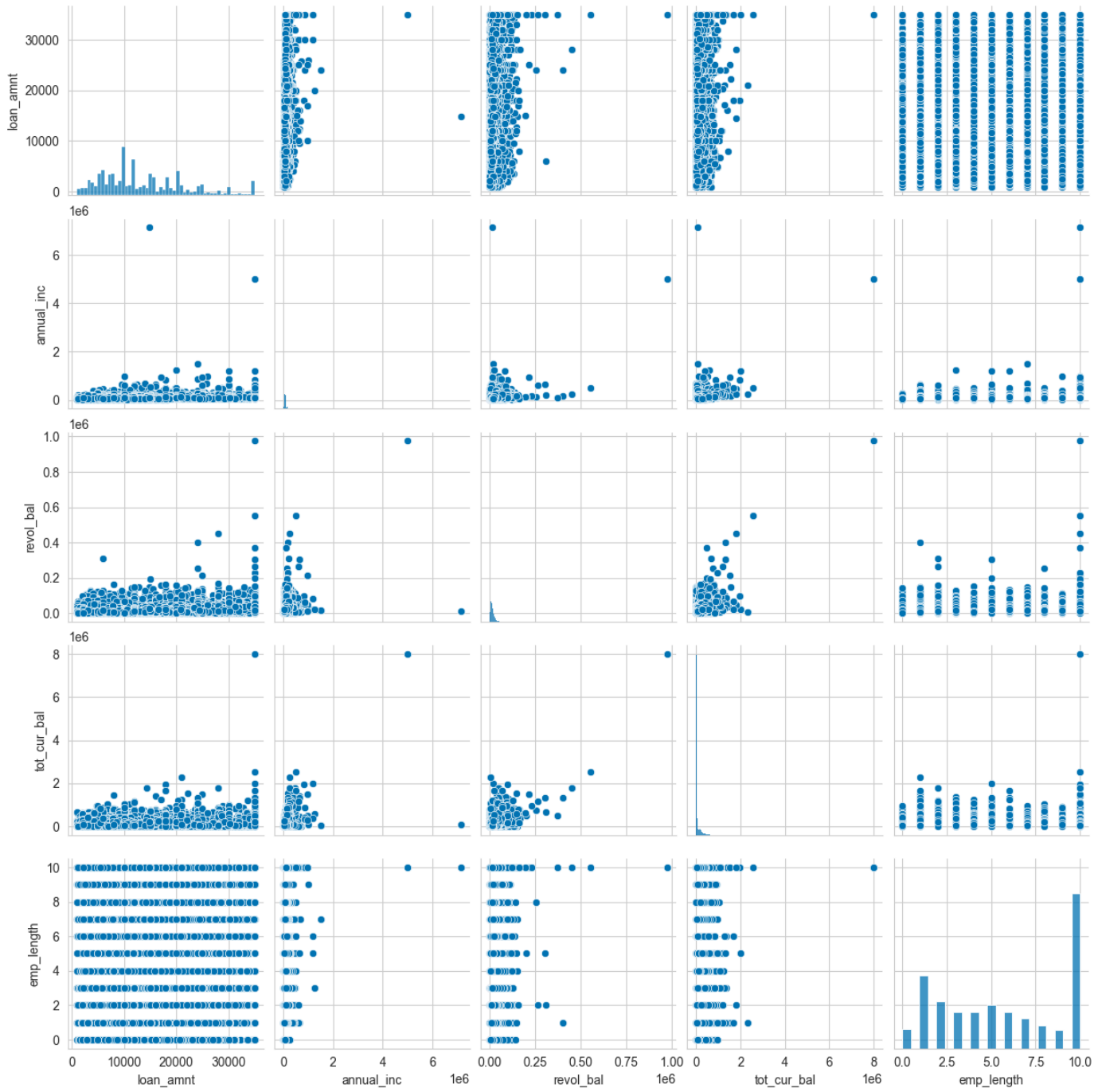


In [5]:

```
#Code Block 04

df_outliers = df_loandata_clean[['loan_amnt','annual_inc', 'revol_bal', 'tot_cur_bal', '
sns.pairplot(df_outliers)
```

Out[5]:

```
<seaborn.axisgrid.PairGrid at 0x7fedf8197280>
```

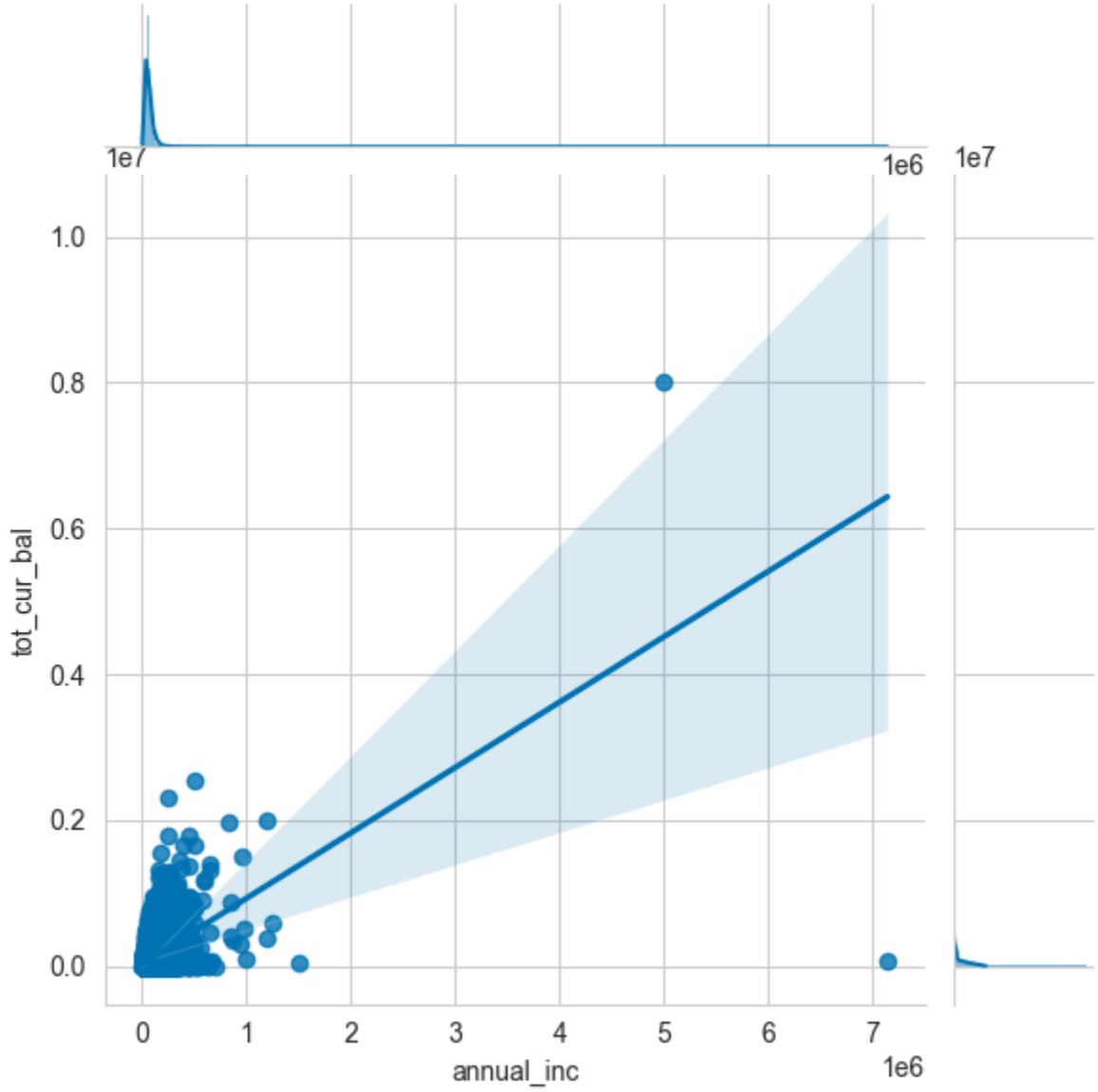


In [6]:

```
#Code Block 05

sns.jointplot(x="annual_inc", y="tot_cur_bal", data=df_loandata_clean, kind='reg')
```

Out[6]:

```
<seaborn.axisgrid.JointGrid at 0x7feeaa1be380>
```



In [7]:

```
#Code Block 06

df_loandata_clean['annual_inc'].value_counts()
```

Out[7]:

```
annual_inc
60000.0    1456
50000.0    1395
40000.0    1165
45000.0    1137
65000.0    1110
           ...
79445.0       1
92200.0       1
70584.0       1
36717.0       1
39990.0       1
Name: count, Length: 4043, dtype: int64
```

In [8]:

```
#Code Block 07

df_annual_inc = df_loandata_clean['annual_inc'].value_counts()
```

```
df_annual_inc = pd.DataFrame(df_annual_inc).reset_index()
print(df_annual_inc.columns)
#df_annual_inc.sort_values(by='index', ascending=False).head(5)
df_annual_inc.sort_values(by='annual_inc', ascending=False).head(5)
```

```
Index(['annual_inc', 'count'], dtype='object')
```

Out[8]:

	annual_inc	count
1913	7141778.0	1
2031	5000000.0	1
3609	1500000.0	1
2413	1250000.0	1
698	1200000.0	2

Note: In the lecture, we used 'index' to sort values after calling reset_index(), as in:

df_annual_inc = df_loandata_clean['annual_inc'].value_counts() df_annual_inc = pd.DataFrame(df_annual_inc).reset_index() df_annual_inc.sort_values(by='index', ascending=False).head(5)

However, depending on the version of pandas or the structure of the resulting DataFrame, the default column name 'index' may not be preserved. For example, after reset_index(), the columns are named something like ['annual_inc', 'count'] instead, as you see above.

To avoid errors, always check the column names using df.columns, and sort using the actual column name (e.g., 'annual_inc') instead of 'index'.

Same output with one line of code

In [9]:

```
#Code Block 08

pd.DataFrame(df_loandata_clean['annual_inc'].value_counts()).reset_index().sort_values(b
```

Out[9]:

	annual_inc	count
1913	7141778.0	1
2031	5000000.0	1
3609	1500000.0	1
2413	1250000.0	1
698	1200000.0	2

Look at the two annual incomes that are > 1000000.0

- Is it worth keeping these six data points?

In [17]:

```
#Code Block 09

df_loandata_clean[df_loandata_clean['annual_inc'] >=1000000]
```

Out[17]:

	member_id	loan_amnt	term	int_rate	annual_inc	delinq_2yrs	inq_last_6mths	mths_since_last
9523	1797692	35000	36	12.12	1200000.0	0	2	
18299	1407521	24000	60	21.00	1500000.0	0	0	
30802	2432841	20000	36	8.90	1250000.0	0	2	
35284	2839463	35000	36	15.31	5000000.0	1	2	
36011	2926819	30000	36	11.14	1200000.0	0	0	
36944	3267055	14825	36	13.11	7141778.0	0	2	

In [28]:

```
#Code Block 10

pd.DataFrame(df_loandata_clean['revol_bal'].value_counts()).reset_index().sort_values(by
```

Out[28]:

	revol_bal	count
13028	975800	1
12409	552758	1
11456	451481	1
11165	401258	1
12433	371817	1
12473	308071	1
12307	303993	1
12446	264260	1
11143	252191	1
12771	229112	1

Look at the two revol_bal that are > 200000

- Is it worth keeping data points above 200,000?

In [29]:

```
#Code Block 11

df_loandata_clean[df_loandata_clean['revol_bal'] >=200000]
```

Out[29]:

	member_id	loan_amnt	term	int_rate	annual_inc	delinq_2yrs	inq_last_6mths	mths_since_last
32964	2730966	25000	36	7.62	954285.00	0	0	

	member_id	loan_amnt	term	int_rate	annual_inc	delinq_2yrs	inq_last_6mths	mths_since_last
33684	2739932	24000	36	6.62	160000.00	0	3	
33749	2740605	24000	60	17.27	141000.00	4	1	
33906	2742178	28000	36	17.77	250000.00	0	5	
34510	2829819	35000	36	18.75	175000.00	0	2	
35284	2839463	35000	36	15.31	5000000.00	1	2	
35982	2917830	35000	60	16.29	650000.00	0	1	
36558	3017184	6000	36	16.29	200000.00	0	0	
36940	3266877	35000	60	16.29	92974.96	0	0	
36982	3376772	35000	36	11.14	600000.00	0	1	
37077	3417245	35000	60	16.29	500000.00	0	1	

In [31]:

```
#Code Block 12

pd.DataFrame(df_loandata_clean['tot_cur_bal'].value_counts()).reset_index().sort_values(
```

Out[31]:

	tot_cur_bal	count
7720	8000078	1
8941	2547166	1
19902	2302431	1
18070	2008009	1
18260	1967533	1
11751	1787296	1
12688	1785763	1
10901	1662481	1
7713	1657520	1
5236	1549458	1

In [32]:

```
df_loandata_clean.head()
```

Out[32]:

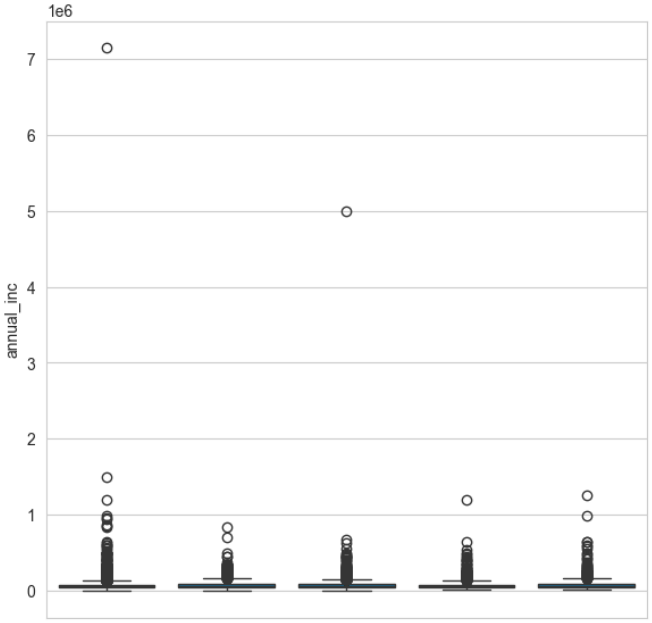
	member_id	loan_amnt	term	int_rate	annual_inc	delinq_2yrs	inq_last_6mths	mths_since_last_deli
0	1581986	9000	36	12.12	45000.0	0	3	
1	1751708	6625	36	11.14	28000.0	1	0	
2	1666916	9800	36	12.12	50000.0	0	0	
3	1758003	4250	36	8.90	38000.0	2	3	
4	1730191	16000	36	7.90	60000.0	0	0	

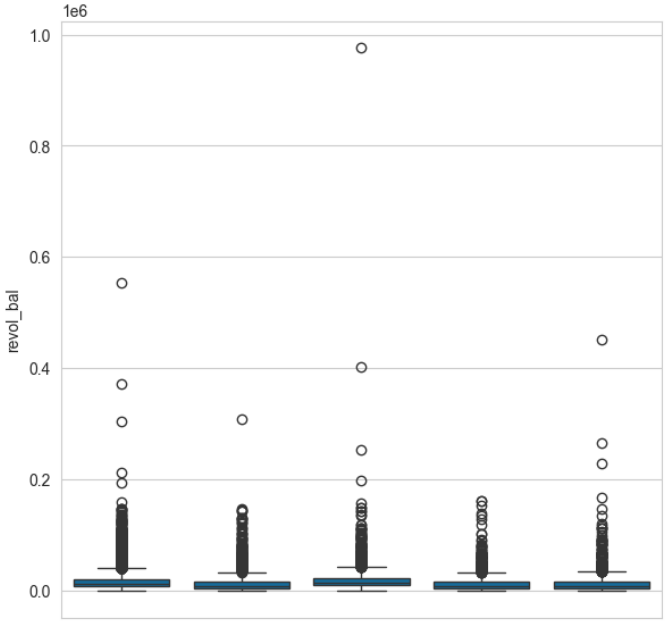
In [33]:

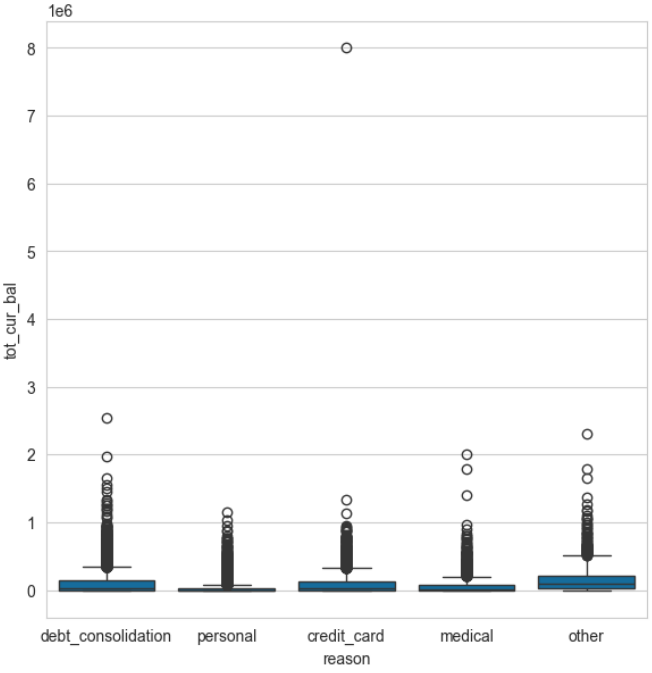
```
#Code Block 13

f, axes = plt.subplots(2, 2, figsize=(15, 15), sharex=True)
sns.boxplot(y='annual_inc', x = 'reason',  data=df_loandata_clean, ax=axes[0,0])
sns.boxplot(y='revol_bal', x = 'reason',  data=df_loandata_clean, ax=axes[0,1])
sns.boxplot(y='tot_cur_bal', x = 'reason',  data=df_loandata_clean, ax=axes[1,0])
sns.boxplot(y='emp_length', x = 'reason',  data=df_loandata_clean, ax=axes[1,1])
```

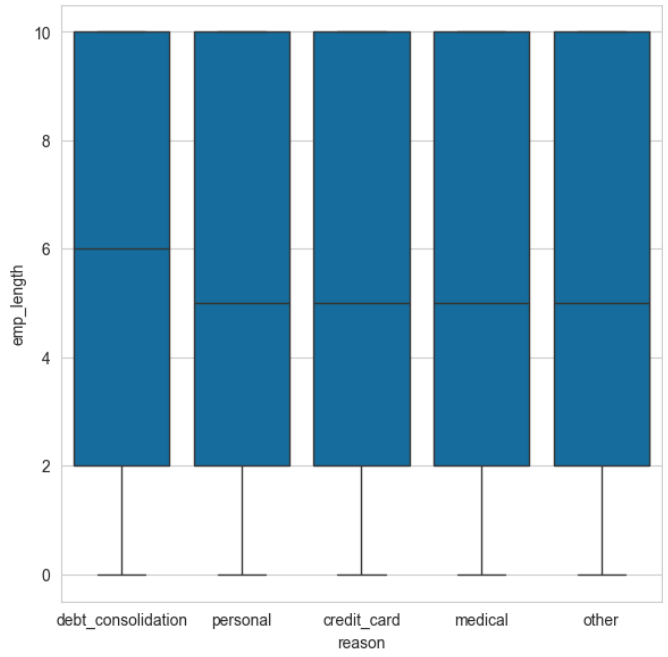
Out[33]:

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









Top

Remove outliers

Why are we removing observations?

- Appleton gives mico loans to individuals that need help with debt consolidation, medical bills, etc. They are normal people that just need a short term loan. Therefore, the objective is not to predict outliers, but instead the average person that asks for a loan. Outliers may skew the results.

Therefore:

- Remove records with annual_inc > 1,000,000
- Remove records with revol_bal > 200,000
- Remove records with total current balance > 2,000,000
- For employee length we change that to categories and then dummy variables

In [34]:
```
#Code Block 14

df_loandata_clean.shape
```

Out[34]:
```
(37132, 27)
```

In [35]:
```
#Code Block 15

df_loandata_clean.drop(df_loandata_clean[df_loandata_clean.annual_inc > 1000000].index,
df_loandata_clean.drop(df_loandata_clean[df_loandata_clean.revol_bal > 200000].index, in

#df_loandata_clean.drop(df_loandata_clean[df_loandata_clean.tot_cur_bal > 2000000].index
df_loandata_clean.shape
```

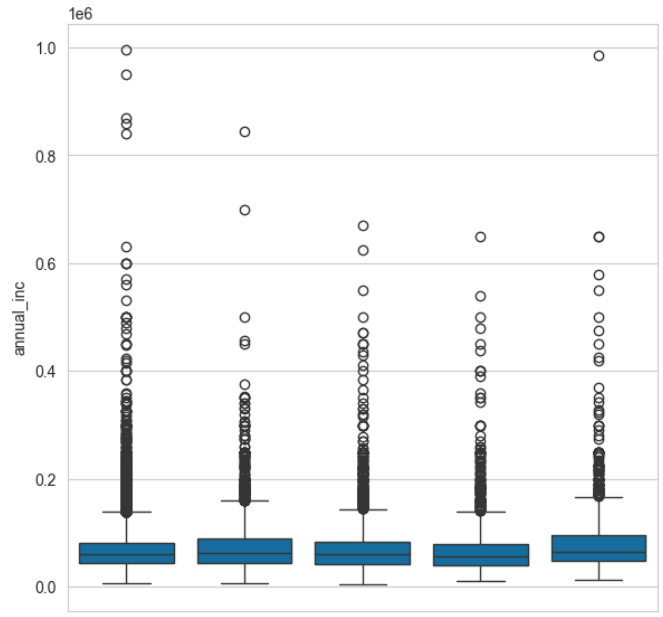
Out[35]:
```
(37116, 27)
```

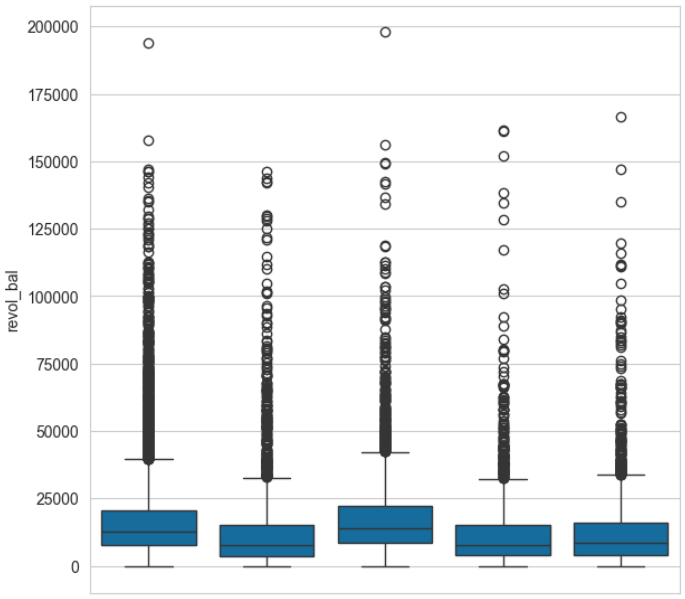
In [36]:
```
#Code Block 16
df_loandata_clean = df_loandata_clean[df_loandata_clean['tot_cur_bal'] < 2000000]
df_loandata_clean.shape
```

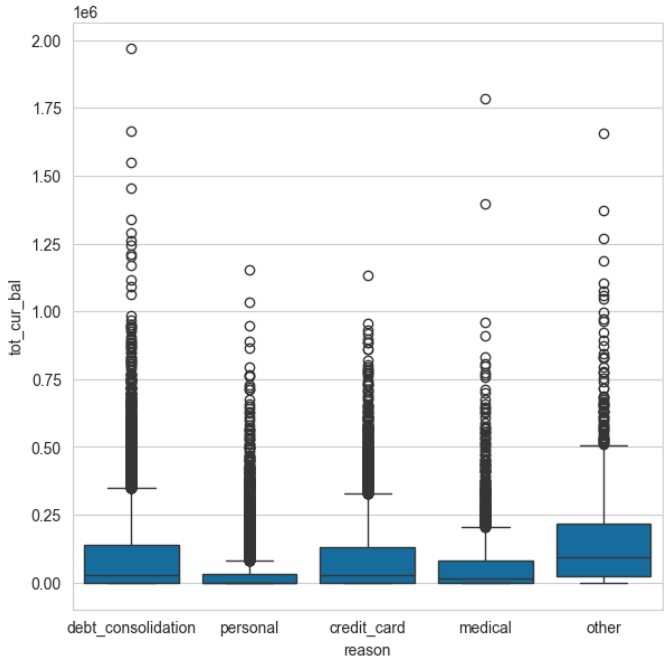
Out[36]:
```
(37115, 27)
```

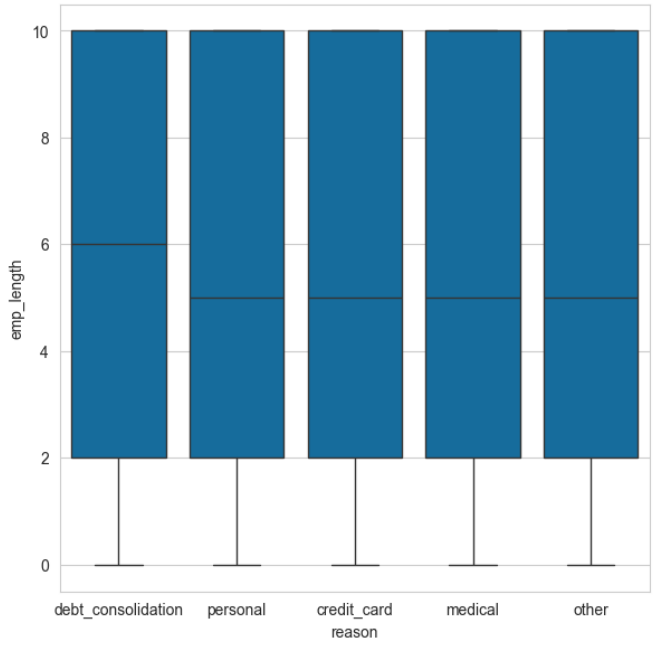
In [37]:
```
#Code Block 17

f, axes = plt.subplots(2, 2, figsize=(15, 15), sharex=True)
sns.boxplot(y='annual_inc', x = 'reason',  data=df_loandata_clean, ax=axes[0,0])
sns.boxplot(y='revol_bal', x = 'reason',  data=df_loandata_clean, ax=axes[0,1])
sns.boxplot(y='tot_cur_bal', x = 'reason',  data=df_loandata_clean, ax=axes[1,0])
sns.boxplot(y='emp_length', x = 'reason',  data=df_loandata_clean, ax=axes[1,1])
```

Out[37]:
```
<Axes: xlabel='reason', ylabel='emp_length'>
```









Re-code the Data

Top

Recode categorical data - label encoder

4 ways to recode categorical data

- **label encoder** - changes categories to integers based on alphabetical order

- **hot one encoder** - changes one column of categorical data into several binary (dummy) columns
- **use a custom function** for changing categories to integers
- **merge** from a separate dataset

In [38]:

```
#Code Block 18

df_loandata_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 37115 entries, 0 to 37131
Data columns (total 27 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   member_id                    37115 non-null  int64
 1   loan_amnt                    37115 non-null  int64
 2   term                         37115 non-null  int64
 3   int_rate                     37115 non-null  float64
 4   annual_inc                   37115 non-null  float64
 5   delinq_2yrs                  37115 non-null  int64
 6   inq_last_6mths               37115 non-null  int64
 7   mths_since_last_delinq       37115 non-null  int64
 8   mths_since_last_record       37115 non-null  int64
 9   open_acc                     37115 non-null  int64
 10  pub_rec                      37115 non-null  int64
 11  revol_bal                    37115 non-null  int64
 12  total_acc                    37115 non-null  int64
 13  total_debt_paid              37115 non-null  float64
 14  princ_int_ratio              37115 non-null  float64
 15  collections_12_mths_ex_med   37115 non-null  int64
 16  mths_since_last_major_derog  37115 non-null  int64
 17  acc_now_delinq               37115 non-null  int64
 18  tot_coll_amt                 37115 non-null  int64
 19  tot_cur_bal                  37115 non-null  int64
 20  total_credit_rv              37115 non-null  int64
 21  revol_util                   37115 non-null  float64
 22  sub_grade                    37115 non-null  object
 23  emp_length                   37115 non-null  int64
 24  home_ownership               37115 non-null  object
 25  loan_status                  37115 non-null  object
 26  reason                       37115 non-null  object
dtypes: float64(5), int64(18), object(4)
memory usage: 7.9+ MB
```

Convert all object features:

- sub_grade - label encoder
- loan_status - merge with new dataset
- reason - one-hot encoding
- home_ownership - one-hot encoding

Convert non-linear features

- emp_length - will be converted to an object with a defined function and then encode with one-hot encoding

Recode using Label Encoding

- Change subgrade to a numerical value and assume it is linear

In [39]:
```
#Code Block 19

from sklearn.preprocessing import LabelEncoder
lc = LabelEncoder()
```

In [40]:
```
#Code Block 20

df_loandata_clean.sub_grade.value_counts()
```

Out[40]:
```
sub_grade
B3    3766
B4    2838
C1    2529
B5    2494
C2    2370
B2    2322
A4    1885
A5    1696
B1    1651
C4    1377
D1    1317
D2    1294
C3    1279
C5    1151
D3    1050
A3    1010
A1    1001
D4     991
A2     896
D5     803
E1     558
E2     522
E4     427
E3     418
E5     336
```

```
F1     286
F2     232
F3     175
F4     131
F5     124
G1      86
G2      51
G3      23
G4      17
G5       9
Name: count, dtype: int64
```

In [41]:

```
#Code Block 21

df_loandata_clean.head()
```

Out[41]:

	member_id	loan_amnt	term	int_rate	annual_inc	delinq_2yrs	inq_last_6mths	mths_since_last_deli
0	1581986	9000	36	12.12	45000.0	0	3	
1	1751708	6625	36	11.14	28000.0	1	0	
2	1666916	9800	36	12.12	50000.0	0	0	
3	1758003	4250	36	8.90	38000.0	2	3	
4	1730191	16000	36	7.90	60000.0	0	0	

In [42]:

```
#Code Block 22

df_loandata_clean['sub_grade'] = lc.fit_transform(df_loandata_clean['sub_grade'])
```

In [43]:

```
#Code Block 23

df_loandata_clean['sub_grade'].value_counts()
```

Out[43]:

```
sub_grade
7     3766
8     2838
10    2529
9     2494
11    2370
6     2322
3     1885
4     1696
5     1651
13    1377
15    1317
16    1294
12    1279
14    1151
17    1050
2     1010
0     1001
18     991
```

```
1      896
19     803
20     558
21     522
23     427
22     418
24     336
25     286
26     232
27     175
28     131
29     124
30      86
31      51
32      23
33      17
34       9
Name: count, dtype: int64
```

Top

Recode Categorical using a Defined Function

In [44]:

```
#Code Block 24

df_loandata_clean['emp_length'].value_counts()
```

Out[44]:

```
emp_length
10    11104
1      5219
2      3359
5      3127
4      2632
3      2590
6      2573
7      2165
8      1635
0      1375
9      1336
Name: count, dtype: int64
```

How to recode?

- 10 refers to 10 or more
- By talking to management, they believe that this can be changed to:
  - **10 or more**: 10
  - **7 to 9**: 7, 8, and 9
  - **4 to 6**: 4, 5, and 6
  - **3 or less**: 0, 1, 2, and 3

In [45]:

```
#Code Block 25

def EmpLength(d):
    if d['emp_length'] == 10:
        return "10 or more"
    elif d['emp_length'] >= 7:
        return "7 to 9"
    elif d['emp_length'] >= 4:
        return "4 to 6"
    else:
        return "3 or less"
df_loandata_clean['emp_length_cat'] = df_loandata_clean.apply(EmpLength, axis = 1)
df_loandata_clean[['emp_length', 'emp_length_cat']].head()
```

Out[45]:

	emp_length	emp_length_cat
0	6	4 to 6
1	3	3 or less
2	10	10 or more
3	4	4 to 6
4	10	10 or more

In [46]:

```
df_loandata_clean.head()
```

Out[46]:

	member_id	loan_amnt	term	int_rate	annual_inc	delinq_2yrs	inq_last_6mths	mths_since_last_deli
0	1581986	9000	36	12.12	45000.0	0	3	
1	1751708	6625	36	11.14	28000.0	1	0	
2	1666916	9800	36	12.12	50000.0	0	0	
3	1758003	4250	36	8.90	38000.0	2	3	
4	1730191	16000	36	7.90	60000.0	0	0	

Top

Recode Categorical using dummy variables

How to automate the dropping of the largest dummy variable

In [47]:

```
#Code Block 26

df_loandata_clean['home_ownership'].value_counts()
```

Out[47]:

```
home_ownership
MORTGAGE    18081
RENT        16096
OWN          2938
Name: count, dtype: int64
```

In [48]:

```
#Code Block 27

df_dummy = pd.DataFrame(df_loandata_clean['home_ownership'].value_counts().reset_index()
df_dummy
```

Out[48]:

	home_ownership	count
0	MORTGAGE	18081
1	RENT	16096
2	OWN	2938

In [49]:

```
#Code Block 28

var_dummy = df_dummy.iloc[0, 0]
var_dummy
```

Out[49]:

```
'MORTGAGE'
```

In [50]:

```
#Code Block 29

var_dummy_prefix = "home"
var_dumpre = var_dummy_prefix + "_" +  var_dummy
var_dumpre
```

Out[50]:

```
'home_MORTGAGE'
```

In [53]:

```
#Code Block 29

#Added  dtype=float to convert to display 0/1 values than the boolean
dummies = pd.get_dummies(df_loandata_clean['home_ownership'], drop_first = False, prefix
dummies.head()
```

Out[53]:

	home_MORTGAGE	home_OWN	home_RENT
0	0.0	0.0	1.0
1	0.0	0.0	1.0
2	0.0	0.0	1.0
3	0.0	1.0	0.0
4	1.0	0.0	0.0

In [54]:

```
#Code Block 30

dummies = dummies.drop(var_dumpre, axis = 1)
dummies.head()
```

Out[54]:

	home_OWN	home_RENT
0	0.0	1.0
1	0.0	1.0
2	0.0	1.0
3	1.0	0.0
4	0.0	0.0

In [55]:

```
#Code Block 32

df_loandata_clean = pd.concat([df_loandata_clean, dummies], axis = 1)
df_loandata_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 37115 entries, 0 to 37131
Data columns (total 30 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   member_id                    37115 non-null  int64
 1   loan_amnt                    37115 non-null  int64
 2   term                         37115 non-null  int64
 3   int_rate                     37115 non-null  float64
 4   annual_inc                   37115 non-null  float64
 5   delinq_2yrs                  37115 non-null  int64
 6   inq_last_6mths               37115 non-null  int64
 7   mths_since_last_delinq       37115 non-null  int64
 8   mths_since_last_record       37115 non-null  int64
 9   open_acc                     37115 non-null  int64
 10  pub_rec                      37115 non-null  int64
 11  revol_bal                    37115 non-null  int64
 12  total_acc                    37115 non-null  int64
 13  total_debt_paid              37115 non-null  float64
 14  princ_int_ratio              37115 non-null  float64
 15  collections_12_mths_ex_med   37115 non-null  int64
 16  mths_since_last_major_derog  37115 non-null  int64
 17  acc_now_delinq               37115 non-null  int64
 18  tot_coll_amt                 37115 non-null  int64
 19  tot_cur_bal                  37115 non-null  int64
 20  total_credit_rv              37115 non-null  int64
 21  revol_util                   37115 non-null  float64
 22  sub_grade                    37115 non-null  int64
 23  emp_length                   37115 non-null  int64
 24  home_ownership               37115 non-null  object
 25  loan_status                  37115 non-null  object
 26  reason                       37115 non-null  object
 27  emp_length_cat               37115 non-null  object
 28  home_OWN                     37115 non-null  float64
 29  home_RENT                    37115 non-null  float64
```

```
dtypes: float64(7), int64(19), object(4)
memory usage: 8.8+ MB
```

Create dummy variables in one code block

In [57]:

```
#Code Block 33

#Create prefix for dummy variables
var_dummy_prefix = "reason"


#Find the highest count category
df_dummy = pd.DataFrame(df_loandata_clean['reason'].value_counts().reset_index())
var_dummy = df_dummy.iloc[0, 0]

#Create variable to drop the highest count column
var_dumpre = var_dummy_prefix + "_" +  var_dummy

#Create dummy variables
dummies = pd.get_dummies(df_loandata_clean['reason'], drop_first = False, prefix=var_dum

#Drop the highest count dummy variable
dummies = dummies.drop(var_dumpre, axis = 1)

#Concat the dummy variables to the main dataset
df_loandata_clean = pd.concat([df_loandata_clean, dummies], axis = 1)

#Display the variables and new dataset

display(df_dummy)
print("----------------------")
print("Highest Count:")
print(var_dummy)
print("----------------------")

display(dummies.head())
df_loandata_clean.head()
```

	reason	count
0	debt_consolidation	22312
1	credit_card	7234
2	personal	3121
3	other	2270
4	medical	2178

```
----------------------
Highest Count:
debt_consolidation
----------------------
```

	reason_credit_card	reason_medical	reason_other	reason_personal
0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	1.0

	reason_credit_card	reason_medical	reason_other	reason_personal
2	1.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	1.0

Out[57]:

	member_id	loan_amnt	term	int_rate	annual_inc	delinq_2yrs	inq_last_6mths	mths_since_last_deli
0	1581986	9000	36	12.12	45000.0	0	3	
1	1751708	6625	36	11.14	28000.0	1	0	
2	1666916	9800	36	12.12	50000.0	0	0	
3	1758003	4250	36	8.90	38000.0	2	3	
4	1730191	16000	36	7.90	60000.0	0	0	

In [58]:

```
#Code Block 34

#Create prefix for dummy variables
var_dummy_prefix = "emp"


#Find the highest count category
df_dummy = pd.DataFrame(df_loandata_clean['emp_length_cat'].value_counts().reset_index()
var_dummy = df_dummy.iloc[0, 0]

#Create variable to drop the highest count column
var_dumpre = var_dummy_prefix + "_" +  var_dummy

#Create dummy variables
dummies = pd.get_dummies(df_loandata_clean['emp_length_cat'], drop_first = False, prefix

#Drop the highest count dummy variable
dummies = dummies.drop(var_dumpre, axis = 1)

#Concat the dummy variables to the main dataset
df_loandata_clean = pd.concat([df_loandata_clean, dummies], axis = 1)

#Display the variables and new dataset

display(df_dummy)
print("----------------------")
print("Highest Count:")
print(var_dummy)
print("----------------------")

display(dummies.head())
df_loandata_clean.head()
```

	emp_length_cat	count
0	3 or less	12543
1	10 or more	11104

	emp_length_cat	count
2	4 to 6	8332
3	7 to 9	5136

```
----------------------
Highest Count:
3 or less
----------------------
```

	emp_10 or more	emp_4 to 6	emp_7 to 9
0	0.0	1.0	0.0
1	0.0	0.0	0.0
2	1.0	0.0	0.0
3	0.0	1.0	0.0
4	1.0	0.0	0.0

Out[58]:

	member_id	loan_amnt	term	int_rate	annual_inc	delinq_2yrs	inq_last_6mths	mths_since_last_deli
0	1581986	9000	36	12.12	45000.0	0	3	
1	1751708	6625	36	11.14	28000.0	1	0	
2	1666916	9800	36	12.12	50000.0	0	0	
3	1758003	4250	36	8.90	38000.0	2	3	
4	1730191	16000	36	7.90	60000.0	0	0	

What to do with Term?

- Term is the number of months for each loan (36 or 60 months)
- Since it is not a linear relationship, it is actually a label (based on numbers), it should be treated as a label/categorical variable.

In [59]:

```
#Code Block 35

df_loandata_clean['term'].value_counts()
```

Out[59]:

```
term
36    30325
60     6790
Name: count, dtype: int64
```

How to code?

- Since most loans are for 36 months, we will want to see if 60 month term loans have an effect on the loan amount.

In [61]:

```
#Code Block 36

#Create prefix for dummy variables
var_dummy_prefix = "term"


#Find the highest count category
df_dummy = pd.DataFrame(df_loandata_clean['term'].value_counts().reset_index())
var_dummy = df_dummy.iloc[0, 0]

#Create variable to drop the highest count column

#NOTE: Since 60 is an integer, you need to set it as a string before you can concatenate
var_dumpre = var_dummy_prefix + "_" +  str(var_dummy)

#Create dummy variables
dummies = pd.get_dummies(df_loandata_clean['term'], drop_first = False, prefix=var_dummy

#Drop the highest count dummy variable
dummies = dummies.drop(var_dumpre, axis = 1)

#Concat the dummy variables to the main dataset
df_loandata_clean = pd.concat([df_loandata_clean, dummies], axis = 1)

#Display the variables and new dataset

display(df_dummy)
print("----------------------")
print("Highest Count:")
print(var_dummy)
print("----------------------")

display(dummies.head())
df_loandata_clean.head()
```

	term	count
0	36	30325
1	60	6790

```
----------------------
Highest Count:
36
----------------------
```

	term_60
0	0.0
1	0.0
2	0.0
3	0.0
4	0.0

Out[61]:

	member_id	loan_amnt	term	int_rate	annual_inc	delinq_2yrs	inq_last_6mths	mths_since_last_deli
0	1581986	9000	36	12.12	45000.0	0	3	
1	1751708	6625	36	11.14	28000.0	1	0	
2	1666916	9800	36	12.12	50000.0	0	0	
3	1758003	4250	36	8.90	38000.0	2	3	
4	1730191	16000	36	7.90	60000.0	0	0	

Top

Recode Categorical using by merging two datasets

In [62]:

```
#Code Block 37
#url = 'https://data63206330.file.core.windows.net/data6320/Loan_is_Bad.csv?sp=rl&st=202
df_loanisbad = pd.read_csv("data/Loan_is_Bad.csv", index_col = None, header = 0)
df_loanisbad
```

Out[62]:

	loan_status	loan_is_bad
0	Charged Off	1
1	Current	0
2	Fully Paid	0
3	In Grace Period	0
4	Late (16-30 days)	0
5	Late (31-120 days)	1
6	Default	1

In [63]:

```
#Code Block 38

df_loandata_clean['loan_status'].value_counts()
```

Out[63]:

```
loan_status
Fully Paid            28510
Charged Off            5502
Current                2775
Late (31-120 days)      213
In Grace Period          87
Late (16-30 days)        22
Default                   6
Name: count, dtype: int64
```

In [64]:

```
#Code Block 39

df_loandata_clean = pd.merge(df_loandata_clean, df_loanisbad, on='loan_status', how='lef
```

In [65]:

```
#Code Block 40

df_loandata_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37115 entries, 0 to 37114
Data columns (total 44 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   member_id                     37115 non-null  int64
 1   loan_amnt                     37115 non-null  int64
 2   term                          37115 non-null  int64
 3   int_rate                      37115 non-null  float64
 4   annual_inc                    37115 non-null  float64
 5   delinq_2yrs                   37115 non-null  int64
 6   inq_last_6mths                37115 non-null  int64
 7   mths_since_last_delinq        37115 non-null  int64
 8   mths_since_last_record        37115 non-null  int64
 9   open_acc                      37115 non-null  int64
 10  pub_rec                       37115 non-null  int64
 11  revol_bal                     37115 non-null  int64
 12  total_acc                     37115 non-null  int64
 13  total_debt_paid               37115 non-null  float64
 14  princ_int_ratio               37115 non-null  float64
 15  collections_12_mths_ex_med    37115 non-null  int64
 16  mths_since_last_major_derog   37115 non-null  int64
 17  acc_now_delinq                37115 non-null  int64
 18  tot_coll_amt                  37115 non-null  int64
 19  tot_cur_bal                   37115 non-null  int64
 20  total_credit_rv               37115 non-null  int64
 21  revol_util                    37115 non-null  float64
 22  sub_grade                     37115 non-null  int64
 23  emp_length                    37115 non-null  int64
 24  home_ownership                37115 non-null  object
 25  loan_status                   37115 non-null  object
 26  reason                        37115 non-null  object
 27  emp_length_cat                37115 non-null  object
 28  home_OWN                      37115 non-null  float64
 29  home_RENT                     37115 non-null  float64
 30  reason_credit_card            37115 non-null  bool
 31  reason_medical                37115 non-null  bool
 32  reason_other                  37115 non-null  bool
 33  reason_personal               37115 non-null  bool
 34  reason_credit_card            37115 non-null  float64
 35  reason_medical                37115 non-null  float64
 36  reason_other                  37115 non-null  float64
 37  reason_personal               37115 non-null  float64
 38  emp_10 or more                37115 non-null  float64
 39  emp_4 to 6                    37115 non-null  float64
 40  emp_7 to 9                    37115 non-null  float64
 41  term_60                       37115 non-null  float64
 42  term_60                       37115 non-null  float64
```

```
 43  loan_is_bad                  37115 non-null  int64
dtypes: bool(4), float64(16), int64(20), object(4)
memory usage: 11.5+ MB
```

In [66]:

```
df_loandata_clean.head()
```

Out[66]:

	member_id	loan_amnt	term	int_rate	annual_inc	delinq_2yrs	inq_last_6mths	mths_since_last_deli
0	1581986	9000	36	12.12	45000.0	0	3	
1	1751708	6625	36	11.14	28000.0	1	0	
2	1666916	9800	36	12.12	50000.0	0	0	
3	1758003	4250	36	8.90	38000.0	2	3	
4	1730191	16000	36	7.90	60000.0	0	0	

The Final Dataset

In [67]:

```
#Code Block 41

df_loandata_clean = df_loandata_clean.drop(['term','emp_length','home_ownership', 'loan_
df_loandata_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37115 entries, 0 to 37114
Data columns (total 38 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   member_id                    37115 non-null  int64
 1   loan_amnt                    37115 non-null  int64
 2   int_rate                     37115 non-null  float64
 3   annual_inc                   37115 non-null  float64
 4   delinq_2yrs                  37115 non-null  int64
 5   inq_last_6mths               37115 non-null  int64
 6   mths_since_last_delinq       37115 non-null  int64
 7   mths_since_last_record       37115 non-null  int64
 8   open_acc                     37115 non-null  int64
 9   pub_rec                      37115 non-null  int64
 10  revol_bal                    37115 non-null  int64
 11  total_acc                    37115 non-null  int64
 12  total_debt_paid              37115 non-null  float64
 13  princ_int_ratio              37115 non-null  float64
 14  collections_12_mths_ex_med   37115 non-null  int64
 15  mths_since_last_major_derog  37115 non-null  int64
 16  acc_now_delinq               37115 non-null  int64
 17  tot_coll_amt                 37115 non-null  int64
 18  tot_cur_bal                  37115 non-null  int64
 19  total_credit_rv              37115 non-null  int64
 20  revol_util                   37115 non-null  float64
 21  sub_grade                    37115 non-null  int64
 22  home_OWN                     37115 non-null  float64
 23  home_RENT                    37115 non-null  float64
 24  reason_credit_card           37115 non-null  bool
```

```
 25  reason_medical               37115 non-null  bool
 26  reason_other                 37115 non-null  bool
 27  reason_personal              37115 non-null  bool
 28  reason_credit_card           37115 non-null  float64
 29  reason_medical               37115 non-null  float64
 30  reason_other                 37115 non-null  float64
 31  reason_personal              37115 non-null  float64
 32  emp_10 or more               37115 non-null  float64
 33  emp_4 to 6                   37115 non-null  float64
 34  emp_7 to 9                   37115 non-null  float64
 35  term_60                      37115 non-null  float64
 36  term_60                      37115 non-null  float64
 37  loan_is_bad                  37115 non-null  int64
dtypes: bool(4), float64(16), int64(18)
memory usage: 9.8 MB
```

In [68]:

```
#Code Block 42

#df_loandata_clean.to_csv('data/Appleton_CleanData.csv')
```

In [ ]:

In [ ]: