Scenario 2 - Part 2

NOTE: here is guide to use markdown - https://www.markdownguide.org/basic-syntax/ (https://www.markdownguide.org/basic-syntax/)

Outline of notebook:

- **C1.S2.Py08 - Understanding .info()**
  - .info()
- **C1.S2.Py09 - Understanding data types**
- **C1.S2.Py10 - Converting objects to dates**
- **C1.S2.Py11 - How to use groupby for categorical data**
- **C1.S2.Py12 - How to pivot data**
- **C1.S2.Py13 - How to melt data**
- **C1.S2.Py14 - Changing and correcting data**
- **C1.S2.Py15 - Describing data and looking for outliers**
  - .describe() and other calculations
- **C1.S2.Py16 - How to slice Data**
  - [ ], .iloc, .loc
- **C1.S2.Py17 - How to filter Data**

In [1]:
```
#Code Block 1

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns



#style options

%matplotlib inline
#if you want graphs to automatically without plt.show

pd.set_option('display.max_columns',500) #allows for up to 500 columns to be displayed when
 viewing a dataframe

plt.style.use('seaborn') #a style that can be used for plots - see style reference above
```

In [2]:
```
#Code Block 2
df = pd.read_csv('data/Scenario2_2.csv', index_col = 0, header=0)
    #DOES NOT set the first column to the index
    # and the top row as the headers
```

Scenario 2 - Part 2

C1.S2.Py08 - View data info and its properties

- .info() Allows you to see each column and its corresponding non-null values and data type.

Documentation - https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.info.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.info.html)

In [3]:
```
#Code Block 3
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30071 entries, 0 to 30070
Data columns (total 21 columns):
Member ID                          30071 non-null int64
Loan ID                            30071 non-null int64
Origination Date                   30071 non-null object
Interest Rate                      30071 non-null float64
Amount Funded                      30071 non-null int64
Total Debt                         30063 non-null float64
Annual Income                      30071 non-null int64
Revolving Accounts                 29983 non-null float64
Total Revolving Credit Line        30059 non-null float64
Term                               30071 non-null int64
Grade                              30071 non-null object
Employee Title                     28396 non-null object
Length of Employment               28990 non-null float64
Home Ownership                     30071 non-null object
Income Verification                18804 non-null object
Loan Purpose                       30071 non-null object
Zip Code of Residence              30071 non-null object
State of Residence                 30071 non-null object
Delinquencies Past 24 Months       4877 non-null float64
Credit Inquires Last 6 Months      30071 non-null int64
Open Accounts                      30071 non-null int64
dtypes: float64(6), int64(7), object(8)
memory usage: 5.0+ MB
```

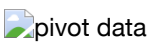
In [4]:
```
# Code Block 4
df.shape
```

Out[4]:
```
(30071, 21)
```

C1.S2.Py09 - Understanding data types

dtypes - in parenthesis () python data type

- object (str or mixed)
  - Text or mixed numeric and non-numeric values
- int64 (int)
  - Integer numbers
- float64 (float)
  - Floating point numbers or continuous numbers (have a decimal point)
- bool (bool)
  - Boolean or True/False values
- datetime64 (NA)
  - Date and time values
- timedelta[ns] (NA)
  - Differences between two datetimes
- category(NA)
  - Finite list of text values



Documentation - [https://pbpython.com/pandas_dtypes.html (https://pbpython.com/pandas_dtypes.html)](https://pbpython.com/pandas_dtypes.html)

Data types for Python variables

In [5]:
```
# Code Block 5
varTerm = 36
print(varTerm)
type(varTerm)
```

```
36
```

Out[5]:
```
int
```

In [6]:
```
# Code Block 6
print(varTerm + 1)
print(varTerm)
```

```
37
36
```

In [7]:
```
# Code Block 7
#varTerm + ' months' #Expect and error - you cannot join a integer and string
```

In [8]:
```
# Code Block 8
str(varTerm) + ' months'
```

Out[8]:
```
'36 months'
```

In [9]:
```
# Code Block 9
varTermString = str(varTerm)
print(type(varTerm))
print(type(varTermString))
```

```
<class 'int'>
<class 'str'>
```

Data types for pandas DataFrames

In [10]:
```
# Code Block 10
df['Interest Rate'].dtypes
```

Out[10]:
```
dtype('float64')
```

In [11]:
```
# Code Block 11
df['Term'].dtypes
# COMMENT: view info with Shift - Tab
```

Out[11]:
```
dtype('int64')
```

In [12]:
```
# Code Block 12
df['Total Debt'].dtypes
```

Out[12]:
```
dtype('float64')
```

In [13]:
```
# Code Block 13
df['TermNum'] = df['Term'] + 1
df.head(2)
```

Out[13]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Total Debt	Annual Income	Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employee Title	Le Empl
0	149512	848058	8/18/19	19.05	7200	154930.0	58000	3874.0	4300.0	36	D	Arkwright	
1	407046	659709	5/21/18	10.16	16000	29116.0	55000	6840.0	24800.0	36	B	School	

In [14]:
```
# Code Block 14
df = df.drop('TermNum', axis = 1) # COMMENT: Drop TermNum since it is not relevant
#df['TermString'] = df['Term'] + " months"
# COMMENT: You cannot join a integer and string, you must convert the integer to a string fi
rst - see Code Block 13
```

In [15]:
```
# Code Block 15
df['TermString'] = df['Term'].astype(str) + " months"
```

In [16]:
```
# Code Block 16
df.head(2)
```

Out[16]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Total Debt	Annual Income	Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employee Title	Le Empl
0	149512	848058	8/18/19	19.05	7200	154930.0	58000	3874.0	4300.0	36	D	Arkwright	
1	407046	659709	5/21/18	10.16	16000	29116.0	55000	6840.0	24800.0	36	B	School	

C1.S2.Py10 - Converting dates from objects

This section changes the columns from an object to a date.

- csv files import a date as an object
- xlsx files import a data as a date (no need to convert)

For every date, you can also create a variable for:

- dayofweek
- month
- year
- day
- dayname
- etc.

In [17]:
```
#Code Block 17
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30071 entries, 0 to 30070
Data columns (total 22 columns):
Member ID                          30071 non-null int64
Loan ID                            30071 non-null int64
Origination Date                   30071 non-null object
Interest Rate                      30071 non-null float64
Amount Funded                      30071 non-null int64
Total Debt                         30063 non-null float64
Annual Income                      30071 non-null int64
Revolving Accounts                 29983 non-null float64
Total Revolving Credit Line        30059 non-null float64
Term                               30071 non-null int64
Grade                              30071 non-null object
Employee Title                     28396 non-null object
Length of Employment               28990 non-null float64
Home Ownership                     30071 non-null object
Income Verification                18804 non-null object
Loan Purpose                       30071 non-null object
Zip Code of Residence              30071 non-null object
State of Residence                 30071 non-null object
Delinquencies Past 24 Months       4877 non-null float64
Credit Inquires Last 6 Months      30071 non-null int64
Open Accounts                      30071 non-null int64
TermString                         30071 non-null object
dtypes: float64(6), int64(7), object(9)
memory usage: 5.3+ MB
```

Import datetime to be able to convert date to elements of a date (month, year, etc.)

https://docs.python.org/3/library/datetime.html (https://docs.python.org/3/library/datetime.html)

In [18]:
```
#Code Block 18
import datetime as dt
```

In [19]:
```
#Code Block 19
df['Origination Date'] = pd.to_datetime(df['Origination Date'])
```

In [20]:
```
#Code Block 20
df['Day'] = df['Origination Date'].dt.dayofweek
# COMMENT: 0 = Monday, 1 = Tuesday, etc.
```

In [21]:
```
#Code Block 21
df['Month'] = df['Origination Date'].dt.month
```

In [22]:
```
#Code Block 22
df['Year'] = df['Origination Date'].dt.year
```

In [23]:
```
#Code Block 23
df.sample(3)
```

Out[23]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Total Debt	Annual Income	Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employee Title
19164	2742158	883465	2019-11-11	18.75	6000	5188.0	40000	958.0	2800.0	36	D	Davita/Tota Renal Care
29707	4016854	805975	2019-04-11	7.90	16000	280453.0	84000	38715.0	76700.0	36	A	Lessie Bates Davis Neighborhood House
18630	2734124	651222	2018-04-22	13.11	24375	33743.0	55000	14706.0	39600.0	60	B	Texco Inc

C1.S2.Py11 - How to use groupby for categorical data

Groupby

https://pandas.pydata.org/pandas-docs/version/0.22/api.html#groupby (https://pandas.pydata.org/pandas-docs/version/0.22/api.html#groupby)

Groupby using COUNT and viewing all columns

In [24]:
```
#Code Block 24
df.groupby('Loan Purpose').count()
```

Out[24]:

Loan Purpose	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Total Debt	Annual Income	Revolving Accounts	Total Revolving Credit Line	Term	Grade	Em
car	351	351	351	351	351	349	351	343	351	351	351	
credit_card	6764	6764	6764	6764	6764	6764	6764	6760	6764	6764	6764	
debt_consolidation	18141	18141	18141	18141	18141	18139	18141	18115	18137	18141	18141	
home_improvement	1440	1440	1440	1440	1440	1439	1440	1423	1439	1440	1440	
house	160	160	160	160	160	160	160	155	160	160	160	
major_purchase	620	620	620	620	620	619	620	614	619	620	620	
medical	296	296	296	296	296	296	296	295	296	296	296	
moving	160	160	160	160	160	160	160	160	160	160	160	
other	1260	1260	1260	1260	1260	1260	1260	1252	1258	1260	1260	
renewable_energy	33	33	33	33	33	33	33	33	33	33	33	
small_business	432	432	432	432	432	431	432	427	430	432	432	
vacation	144	144	144	144	144	143	144	140	142	144	144	
wedding	270	270	270	270	270	270	270	266	270	270	270	

Groupby using MEAN and viewing one column

In [25]:
```
#Code Block 25
df.groupby('Loan Purpose')['Amount Funded'].mean()
```

Out[25]:
```
Loan Purpose
car                    7687.891738
credit_card           13979.017593
debt_consolidation    15043.152252
home_improvement      13325.069444
house                 15823.593750
major_purchase         8695.403226
medical                8169.087838
moving                 6754.375000
other                  9495.654762
renewable_energy      10249.242424
small_business        15773.726852
vacation               5922.048611
wedding               10949.259259
Name: Amount Funded, dtype: float64
```

Groupby using MEDIAN and viewing one column as a DataFrame

In [26]:
```
#Code Block 26
df_loanpurpose = df.groupby('Loan Purpose')['Amount Funded'].median()
df_loanpurpose = pd.DataFrame(df_loanpurpose)
df_loanpurpose
```

Out[26]:

Loan Purpose	Amount Funded
car	6275.0
credit_card	12000.0
debt_consolidation	14000.0
home_improvement	10800.0
house	14000.0
major_purchase	6500.0
medical	6412.5
moving	5312.5
other	7500.0
renewable_energy	8000.0
small_business	13862.5
vacation	5000.0
wedding	9437.5

Groupby using MAX and viewing one column as a DataFrame and resetting index

In [27]:
```
#Code Block 27
df_loanpurpose2 = df.groupby('Loan Purpose')['Amount Funded'].min()
df_loanpurpose2 = pd.DataFrame(df_loanpurpose2)
df_loanpurpose2 = df_loanpurpose2.reset_index()
df_loanpurpose2
```

Out[27]:

	Loan Purpose	Amount Funded
0	car	1000
1	credit_card	1000
2	debt_consolidation	1000
3	home_improvement	1000
4	house	1400
5	major_purchase	1000
6	medical	1000
7	moving	1000
8	other	1000
9	renewable_energy	2000
10	small_business	1000
11	vacation	1000
12	wedding	1000

Groupby with TWO columns using MEAN and viewing one column

In [28]:
```
#Code Block 28
df.groupby(['Loan Purpose', 'Home Ownership'])['Amount Funded'].mean()
```

Out[28]:

Loan Purpose	Home Ownership	
car	MORTGAGE	7975.000000
	OWN	6811.979167
	RENT	7612.500000
credit_card	MORGTAGE	7500.000000
	MORTGAGE	15682.588886
	NONE	13426.086957
	OTHER	13636.458333
	OWN	13249.754420
	RENT	12124.737762
debt_consolidation	MORTGAGE	16663.936936
	NONE	20119.444444
	OTHER	16185.714286
	OWN	13695.715962
	RENT	13263.350577
home_improvement	MORGTAGE	10000.000000
	MORTGAGE	13912.195652
	NONE	13650.000000
	OTHER	8181.250000
	OWN	10499.852071
	RENT	11778.097345
house	MORTGAGE	18556.779661
	OWN	15063.043478
	RENT	13980.448718
major_purchase	MORTGAGE	8561.693548
	OWN	9286.500000
	RENT	8741.153846
medical	MORTGAGE	8780.608974
	OWN	8000.000000
	RENT	7353.828829
moving	MORTGAGE	10255.714286
	OWN	7817.857143
	RENT	5516.216216
other	MORTGAGE	10409.530387
	OTHER	6625.000000
	OWN	10045.370370
	RENT	8586.554276
renewable_energy	MORTGAGE	11576.666667
	OWN	8580.000000
	RENT	9359.615385
small_business	MORTGAGE	17460.426009
	OTHER	34475.000000
	OWN	15057.812500
	RENT	13660.511364
vacation	MORTGAGE	6157.547170
	OWN	5573.750000
	RENT	5844.366197
wedding	MORTGAGE	12236.778846
	OWN	11798.611111
	RENT	9941.216216

Name: Amount Funded, dtype: float64

Groupby with two columns using mean and viewing one column as a DataFrame

In [29]:
```
#Code Block 29
df_loanown = df.groupby(['Loan Purpose', 'Home Ownership'])['Amount Funded'].mean()
df_loanown = pd.DataFrame(df_loanown).reset_index()
df_loanown
```

Out[29]:

	Loan Purpose	Home Ownership	Amount Funded
0	car	MORTGAGE	7975.000000
1	car	OWN	6811.979167
2	car	RENT	7612.500000
3	credit_card	MORGTAGE	7500.000000
4	credit_card	MORTGAGE	15682.588886
5	credit_card	NONE	13426.086957
6	credit_card	OTHER	13636.458333
7	credit_card	OWN	13249.754420
8	credit_card	RENT	12124.737762
9	debt_consolidation	MORTGAGE	16663.936936
10	debt_consolidation	NONE	20119.444444
11	debt_consolidation	OTHER	16185.714286
12	debt_consolidation	OWN	13695.715962
13	debt_consolidation	RENT	13263.350577
14	home_improvement	MORGTAGE	10000.000000
15	home_improvement	MORTGAGE	13912.195652
16	home_improvement	NONE	13650.000000
17	home_improvement	OTHER	8181.250000
18	home_improvement	OWN	10499.852071
19	home_improvement	RENT	11778.097345
20	house	MORTGAGE	18556.779661
21	house	OWN	15063.043478
22	house	RENT	13980.448718
23	major_purchase	MORTGAGE	8561.693548
24	major_purchase	OWN	9286.500000
25	major_purchase	RENT	8741.153846
26	medical	MORTGAGE	8780.608974
27	medical	OWN	8000.000000
28	medical	RENT	7353.828829
29	moving	MORTGAGE	10255.714286
30	moving	OWN	7817.857143
31	moving	RENT	5516.216216
32	other	MORTGAGE	10409.530387
33	other	OTHER	6625.000000
34	other	OWN	10045.370370
35	other	RENT	8586.554276
36	renewable_energy	MORTGAGE	11576.666667
37	renewable_energy	OWN	8580.000000
38	renewable_energy	RENT	9359.615385
39	small_business	MORTGAGE	17460.426009
40	small_business	OTHER	34475.000000
41	small_business	OWN	15057.812500
42	small_business	RENT	13660.511364

	Loan Purpose	Home Ownership	Amount Funded
43	vacation	MORTGAGE	6157.547170
44	vacation	OWN	5573.750000
45	vacation	RENT	5844.366197
46	wedding	MORTGAGE	12236.778846
47	wedding	OWN	11798.611111
48	wedding	RENT	9941.216216

In [30]:
```
#Code Block 30
df.groupby(['Home Ownership'])['Home Ownership'].count()
```

Out[30]:
```
Home Ownership
MORGTAGE        2
MORTGAGE    15482
NONE           35
OTHER          37
OWN          2303
RENT        12212
Name: Home Ownership, dtype: int64
```

NOTE:

We will address the misspellings of MORGTAGE in another video (**C1.S4.Py05- Changing and correcting data**)

C1.S2.Py12 - How to pivot data

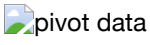
- View the sum of all loans per year and month based on reason for leaving

images from: https://pandas.pydata.org/pandas-docs/stable/user_guide/reshaping.html (https://pandas.pydata.org/pandas-docs/stable/user_guide/reshaping.html)

Pivot

- Take one column and expand it multiple columns
- pivot = df.pivot_table(index='index', columns='categorical value', values='numerical value')
- set index as an identifier
- set the columns to the column that has the different categories that will be your headers
- set the values to a specific numerical column

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.pivot.html (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.pivot.html)



Melt

- set dataset
- set variables to include that are not melted - id_vars
- melt variable - value_vars https://pandas.pydata.org/pandas-docs/stable/generated/pandas.melt.html (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.melt.html)

In [31]:
```
#Code Block 31
df.head()
```

Out[31]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Total Debt	Annual Income	Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employee Title	L Emp
0	149512	848058	2019-08-18	19.05	7200	154930.0	58000	3874.0	4300.0	36	D	Arkwright	
1	407046	659709	2018-05-21	10.16	16000	29116.0	55000	6840.0	24800.0	36	B	School	
2	507531	601368	2018-01-01	10.16	35000	60019.0	130000	23025.0	55800.0	36	B	gSEMI	
3	513904	761341	2019-01-03	6.03	21000	37603.0	120000	18641.0	85031.0	36	A	Fidelity Investments	
4	603349	885844	2019-11-17	16.29	15000	227890.0	72000	11702.0	26300.0	36	C	NaN	

Pivot data

- Loan Purpose as the rows
- Home Ownership as the columns
- Amount Funded average as the values

In [32]:
```
#Code Block 32
df_pivot = df.pivot_table(index='Loan Purpose', columns='Home Ownership', values='Amount Funded')
df_pivot
```

Out[32]:

Home Ownership	MORGTAGE	MORTGAGE	NONE	OTHER	OWN	RENT
Loan Purpose						
car	NaN	7975.000000	NaN	NaN	6811.979167	7612.500000
credit_card	7500.0	15682.588886	13426.086957	13636.458333	13249.754420	12124.737762
debt_consolidation	NaN	16663.936936	20119.444444	16185.714286	13695.715962	13263.350577
home_improvement	10000.0	13912.195652	13650.000000	8181.250000	10499.852071	11778.097345
house	NaN	18556.779661	NaN	NaN	15063.043478	13980.448718
major_purchase	NaN	8561.693548	NaN	NaN	9286.500000	8741.153846
medical	NaN	8780.608974	NaN	NaN	8000.000000	7353.828829
moving	NaN	10255.714286	NaN	NaN	7817.857143	5516.216216
other	NaN	10409.530387	NaN	6625.000000	10045.370370	8586.554276
renewable_energy	NaN	11576.666667	NaN	NaN	8580.000000	9359.615385
small_business	NaN	17460.426009	NaN	34475.000000	15057.812500	13660.511364
vacation	NaN	6157.547170	NaN	NaN	5573.750000	5844.366197
wedding	NaN	12236.778846	NaN	NaN	11798.611111	9941.216216

Pivot data and reset index

- Loan Purpose as the rows
- Home Ownership as the columns
- Amount Funded average as the values

In [33]:
```
#Code Block 33
df_pivot = df_pivot.reset_index()
df_pivot.head()
```

Out[33]:

Home Ownership	Loan Purpose	MORGTAGE	MORTGAGE	NONE	OTHER	OWN	RENT
0	car	NaN	7975.000000	NaN	NaN	6811.979167	7612.500000
1	credit_card	7500.0	15682.588886	13426.086957	13636.458333	13249.754420	12124.737762
2	debt_consolidation	NaN	16663.936936	20119.444444	16185.714286	13695.715962	13263.350577
3	home_improvement	10000.0	13912.195652	13650.000000	8181.250000	10499.852071	11778.097345
4	house	NaN	18556.779661	NaN	NaN	15063.043478	13980.448718

C1.S2.Py13 - How to melt data

Melt Data

- Create a list of column names of the columns that you want to melt/merge into one column.
  - MORTGAGE, NONE, OTHER, OWN, and RENT
- Melt the data by:
  - Specify the dataset - ***df_columns***.
  - Specify LOAN PURPOSE as the *id_vars*.
  - Taking 5 columns (***MORTGAGE, NONE, OTHER, OWN,*** and ***RENT***) into one column by specifying column names for *value_vars* by using the ***df_pivot_columns***.

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.melt.html (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.melt.html)

In [34]:
```
#Code Block 34
df_pivot_columns = df_pivot.columns
df_pivot_columns
# COMMENT: This creates the list of column names for value_vars in the melt function
```

Out[34]:
```
Index(['Loan Purpose', 'MORGTAGE', 'MORTGAGE', 'NONE', 'OTHER', 'OWN', 'RENT'], dtype='obje
ct', name='Home Ownership')
```

In [35]:
```
#Code Block 35
df_pivot_columns = df_pivot_columns.drop('Loan Purpose')
df_pivot_columns
# COMMENT: Drop Loan Purpose, since this is not one of the columns that should be melted int
o one column.
```

Out[35]:
```
Index(['MORGTAGE', 'MORTGAGE', 'NONE', 'OTHER', 'OWN', 'RENT'], dtype='object', name='Home
Ownership')
```

In [36]:
```
#Code Block 36
df_melt =pd.melt(df_pivot, id_vars=['Loan Purpose'], value_vars=df_pivot_columns)
df_melt.head(15)
```

Out[36]:

	Loan Purpose	Home Ownership	value
0	car	MORGTAGE	NaN
1	credit_card	MORGTAGE	7500.000000
2	debt_consolidation	MORGTAGE	NaN
3	home_improvement	MORGTAGE	10000.000000
4	house	MORGTAGE	NaN
5	major_purchase	MORGTAGE	NaN
6	medical	MORGTAGE	NaN
7	moving	MORGTAGE	NaN
8	other	MORGTAGE	NaN
9	renewable_energy	MORGTAGE	NaN
10	small_business	MORGTAGE	NaN
11	vacation	MORGTAGE	NaN
12	wedding	MORGTAGE	NaN
13	car	MORTGAGE	7975.000000
14	credit_card	MORTGAGE	15682.588886

In [37]:
```
#Code Block 37
df_melt.sort_values(by=['Loan Purpose'], ascending = True).head(15)
```

Out[37]:

	Loan Purpose	Home Ownership	value
0	car	MORGTAGE	NaN
26	car	NONE	NaN
52	car	OWN	6811.979167
13	car	MORTGAGE	7975.000000
65	car	RENT	7612.500000
39	car	OTHER	NaN
1	credit_card	MORGTAGE	7500.000000
53	credit_card	OWN	13249.754420
66	credit_card	RENT	12124.737762
40	credit_card	OTHER	13636.458333
27	credit_card	NONE	13426.086957
14	credit_card	MORTGAGE	15682.588886
2	debt_consolidation	MORGTAGE	NaN
28	debt_consolidation	NONE	20119.444444
54	debt_consolidation	OWN	13695.715962

In [38]:
```
#Code Block 38
df_melt=df_melt.rename(columns = {'value': 'Amount Funded'})
df_melt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 78 entries, 0 to 77
Data columns (total 3 columns):
Loan Purpose      78 non-null object
Home Ownership    78 non-null object
Amount Funded     49 non-null float64
dtypes: float64(1), object(2)
memory usage: 2.0+ KB
```

In [39]:
```
#Code Block 39
df_melt = df_melt.dropna(subset=['Amount Funded'])
df_melt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 49 entries, 1 to 77
Data columns (total 3 columns):
Loan Purpose      49 non-null object
Home Ownership    49 non-null object
Amount Funded     49 non-null float64
dtypes: float64(1), object(2)
memory usage: 1.5+ KB
```

In [40]:
```
#Code Block 40
df_melt.head(15)
```

Out[40]:

	Loan Purpose	Home Ownership	Amount Funded
1	credit_card	MORGTAGE	7500.000000
3	home_improvement	MORGTAGE	10000.000000
13	car	MORTGAGE	7975.000000
14	credit_card	MORTGAGE	15682.588886
15	debt_consolidation	MORTGAGE	16663.936936
16	home_improvement	MORTGAGE	13912.195652
17	house	MORTGAGE	18556.779661
18	major_purchase	MORTGAGE	8561.693548
19	medical	MORTGAGE	8780.608974
20	moving	MORTGAGE	10255.714286
21	other	MORTGAGE	10409.530387
22	renewable_energy	MORTGAGE	11576.666667
23	small_business	MORTGAGE	17460.426009
24	vacation	MORTGAGE	6157.547170
25	wedding	MORTGAGE	12236.778846

C1.S2.Py14 - Changing and correcting data

NOTE: This video addresses the misspellings of MORGTAGE in **C1.S4.Py04a - Using groupby for categorical data**

In [41]:
```
#Code Block 41
df.groupby(['Home Ownership'])['Home Ownership'].count()
```

Out[41]:
```
Home Ownership
MORGTAGE        2
MORTGAGE    15482
NONE           35
OTHER          37
OWN          2303
RENT        12212
Name: Home Ownership, dtype: int64
```

In [42]:
```
#Code Block 42
df[df['Home Ownership'] == 'MORGTAGE' ]
```

Out[42]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Total Debt	Annual Income	Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employee Title	Le Empl
6	735990	789789	2019-02-17	7.62	7500	265809.0	92000	6419.0	43000.0	36	A	TD Bank	
8	778284	746115	2018-12-13	6.03	10000	152402.0	108000	4653.0	46100.0	36	A	FlightStats, Inc.	

In [43]:
```
#Code Block 43
df = df.replace('MORGTAGE', 'MORTGAGE')
df.groupby(['Home Ownership'])['Home Ownership'].count()
```

Out[43]:
```
Home Ownership
MORTGAGE    15484
NONE           35
OTHER          37
OWN          2303
RENT        12212
Name: Home Ownership, dtype: int64
```

In [44]:
```
#Code Block 44
df_openacc = df.copy()
df_openacc = df_openacc[df_openacc['Open Accounts']>20]
print(df_openacc['Open Accounts'].count())
df_openacc.groupby(['Open Accounts'])['Open Accounts'].count()
```

```
1024
```

Out[44]:
```
Open Accounts
21    307
22    209
23    163
24    127
25     74
26     34
27     25
28     18
29     15
30      9
31     12
32      4
33      3
34      6
35      5
37      2
38      1
39      4
43      1
44      1
46      1
50      1
52      1
53      1
Name: Open Accounts, dtype: int64
```

In [45]:
```
#Code Block 45
df_openacc[df_openacc['Open Accounts']>30] = 30
df_openacc.groupby(['Open Accounts'])['Open Accounts'].count()
```

Out[45]:
```
Open Accounts
21    307
22    209
23    163
24    127
25     74
26     34
27     25
28     18
29     15
30     52
Name: Open Accounts, dtype: int64
```

C1.S2.Py15 - Describing data and looking for outliers

Describing data

- .describe()
- other types of calculations can be found at https://pandas.pydata.org/pandas-docs/stable/reference/frame.html (https://pandas.pydata.org/pandas-docs/stable/reference/frame.html) and look for the Computations / descriptive statistics heading

In [46]:
```
#Code Block 46
df.describe()
```

Out[46]:

	Member ID	Loan ID	Interest Rate	Amount Funded	Total Debt	Annual Income	Revolving Accounts	Total Revolving Credit Line
count	3.007100e+04	30071.000000	30071.000000	30071.000000	3.006300e+04	3.007100e+04	2.998300e+04	3.005900e+04
mean	2.558148e+06	758572.627814	13.761040	14089.528117	1.365667e+05	7.378315e+04	1.685330e+04	2.987683e+04
std	8.106320e+05	91431.818948	4.179612	8045.157332	1.607559e+05	8.012846e+04	2.107094e+04	3.007873e+04
min	1.495120e+05	600947.000000	6.000000	1000.000000	5.000000e+00	7.200000e+03	4.000000e+00	1.000000e+02
25%	1.837828e+06	679267.000000	11.140000	8000.000000	2.660150e+04	4.500000e+04	7.438000e+03	1.410000e+04
50%	2.287587e+06	758139.000000	14.090000	12000.000000	7.526200e+04	6.300000e+04	1.287600e+04	2.320000e+04
75%	3.408782e+06	838127.000000	16.290000	19750.000000	2.069760e+05	8.800000e+04	2.152350e+04	3.724550e+04
max	4.076727e+06	917123.000000	24.890000	35000.000000	8.000078e+06	7.141778e+06	1.743266e+06	2.013133e+06

In [ ]:
```

```

In [47]:
```
#Code Block 47
df['Amount Funded'].describe()
```

Out[47]:
```
count    30071.000000
mean     14089.528117
std       8045.157332
min       1000.000000
25%       8000.000000
50%      12000.000000
75%      19750.000000
max      35000.000000
Name: Amount Funded, dtype: float64
```

In [48]:
```
#Code Block 48
AmountFundedStats = df['Amount Funded'].describe().reset_index()
AmountFundedStats
```

Out[48]:

	index	Amount Funded
0	count	30071.000000
1	mean	14089.528117
2	std	8045.157332
3	min	1000.000000
4	25%	8000.000000
5	50%	12000.000000
6	75%	19750.000000
7	max	35000.000000

In [49]:
```
#Code Block 49
df['Amount Funded'].mean()
```

Out[49]:
```
14089.528116790263
```

In [50]:
```
#Code Block 50
df['Amount Funded'].median()
```

Out[50]:
```
12000.0
```

In [51]:
```
#Code Block 51
df['Amount Funded'].std()
```

Out[51]:
```
8045.15733158739
```

In [52]:
```
#Code Block 52
print('The Standard Deviation is ' + str(df['Amount Funded'].std()))
```

```
The Standard Deviation is 8045.15733158739
```

In [53]:
```
#Code Block 53
print ('---------------------------------')
print('The Mean is:')
print(df['Amount Funded'].mean())
print ('---------------------------------')
print('The Median is:')
print(df['Amount Funded'].median())
print ('---------------------------------')
print('The Standard Deviation is:')
print(df['Amount Funded'].std())
print ('---------------------------------')
print('The Variance is:')
print(round(df['Amount Funded'].var(), 2))
print ('---------------------------------')
print('The Skewness is:')
print(round(df['Amount Funded'].skew(), 4))
print ('---------------------------------')
```

```
---------------------------------
The Mean is:
14089.528116790263
---------------------------------
The Median is:
12000.0
---------------------------------
The Standard Deviation is:
8045.15733158739
---------------------------------
The Variance is:
64724556.49
---------------------------------
The Skewness is:
0.7313
---------------------------------
```

C1.S2.Py16 - Looking for outliers

In [54]:
```
#Code Block 54
df['Loan Purpose'].value_counts()
```

Out[54]:
```
debt_consolidation    18141
credit_card            6764
home_improvement       1440
other                  1260
major_purchase          620
small_business          432
car                     351
medical                 296
wedding                 270
house                   160
moving                  160
vacation                144
renewable_energy         33
Name: Loan Purpose, dtype: int64
```

In [57]:
```
#Code Block 55
df['Home Ownership'].value_counts(normalize=True)
```

Out[57]:
```
MORTGAGE    0.514915
RENT        0.406106
OWN         0.076585
OTHER       0.001230
NONE        0.001164
Name: Home Ownership, dtype: float64
```

In [58]:
```
#Code Block 56
df[df['Annual Income'] > 1000000 ].sort_values('Annual Income', ascending=False)
```

Out[58]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Total Debt	Annual Income	Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employe Titl
22304	3267055	726952	2018-11-14	13.11	14825	69548.0	7141778	11351.0	16000.0	36	B	Us post servic
29333	3979312	648723	2018-04-16	12.12	30000	329254.0	6100000	15219.0	16100.0	36	B	BOONE AN SON JEWELER
20416	2839463	742564	2018-12-09	15.31	35000	8000078.0	5000000	975800.0	988000.0	36	C	VISIU asse managemer
29607	3983192	677675	2018-07-15	15.80	35000	1528010.0	2000000	694615.0	757500.0	60	C	R Marke &Sons In
82	1407521	609446	2018-01-11	21.00	24000	49281.0	1500000	18544.0	24000.0	60	E	sfp
21507	2977671	643050	2018-04-05	13.11	18000	66705.0	1350000	17148.0	34200.0	36	B	Argus Healt Stystem In
16408	2432841	855970	2019-09-16	8.90	20000	575475.0	1250000	22427.0	70700.0	36	A	PACAF PM
18028	2726748	814579	2019-04-29	19.72	35000	75069.0	1250000	11419.0	18600.0	60	D	BA System
4773	1797692	761520	2019-01-03	12.12	35000	2008009.0	1200000	20228.0	42300.0	36	B	Highbridg Capit
21110	2926819	619179	2018-01-23	11.14	30000	375355.0	1200000	81138.0	105400.0	36	B	Murray' Chees
27676	3876448	737805	2018-12-02	8.90	21250	221218.0	1200000	12367.0	20900.0	36	A	UP

In [60]:
```
#Code Block 48
df[df['Amount Funded'] == 35000 ].sort_values('Annual Income', ascending=False).head(15)
```

Out[60]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Total Debt	Annual Income	Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employe Titl
20416	2839463	742564	2018-12-09	15.31	35000	8000078.0	5000000	975800.0	988000.0	36	C	VISIUI asse managemer
29607	3983192	677675	2018-07-15	15.80	35000	1528010.0	2000000	694615.0	757500.0	60	C	R Marke &Sons In
18028	2726748	814579	2019-04-29	19.72	35000	75069.0	1250000	11419.0	18600.0	60	D	BA System
4773	1797692	761520	2019-01-03	12.12	35000	2008009.0	1200000	20228.0	42300.0	36	B	Highbridg Capit
4070	1787504	825611	2019-06-12	20.49	35000	862656.0	845000	51228.0	241300.0	60	E	Na
21691	3016955	881434	2019-11-07	15.80	35000	4772549.0	800000	54511.0	62404.0	36	C	Standar Chartere Ban
25822	3777698	853904	2019-09-09	21.00	35000	675477.0	729368	92841.0	101900.0	60	E	DA Davidso
21080	2917830	617029	2018-01-21	16.29	35000	1324144.0	650000	303993.0	334500.0	60	C	William Bla
29558	3982604	689878	2018-08-26	12.12	35000	771617.0	650000	33262.0	42300.0	36	B	Deloitt
22384	3376772	701085	2018-09-30	11.14	35000	1175696.0	600000	264260.0	344900.0	36	B	University Sout Florid
2101	1754958	616050	2018-01-21	12.12	35000	342006.0	500000	6390.0	32300.0	60	B	Goldma Sach
22837	3417245	668938	2018-06-20	16.29	35000	2547166.0	500000	552758.0	645800.0	60	C	Ban
14197	2126967	716798	2018-10-29	18.75	35000	596038.0	500000	32559.0	41663.0	36	D	Na
27079	3869263	739452	2018-12-04	15.31	35000	120995.0	500000	12876.0	20100.0	36	C	Infiniti Hom Loans In
17108	2627993	683794	2018-07-30	10.16	35000	92612.0	500000	48119.0	157900.0	36	B	Na

In [61]:
```
#Code Block 58
df[df['Amount Funded'] == 35000 ].shape
```

Out[61]: (853, 25)

In [62]:
```
#Code Block 59
df[df['Amount Funded'] == 35000 ]['Amount Funded'].count()
```

Out[62]: 853

C1.S2.Py17 - Slice and Filter Data

- https://pandas.pydata.org/pandas-docs/stable/reference/frame.html (https://pandas.pydata.org/pandas-docs/stable/reference/frame.html) and look for Indexing, iteration

In [63]:
```
#Code Block 51
df.head(2)
```

Out[63]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Total Debt	Annual Income	Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employee Title	Le Empl
0	149512	848058	2019-08-18	19.05	7200	154930.0	58000	3874.0	4300.0	36	D	Arkwright	
1	407046	659709	2018-05-21	10.16	16000	29116.0	55000	6840.0	24800.0	36	B	School	

In [64]:
```
#Code Block 52
df['Member ID'].head(2)
```

Out[64]:
```
0    149512
1    407046
Name: Member ID, dtype: int64
```

In [65]:
```
#Code Block 53
df[['Member ID']].head(2)
```

Out[65]:

	Member ID
0	149512
1	407046

In [66]:
```
#Code Block 54
df[['Member ID', 'Loan ID','Interest Rate', 'Amount Funded']].head(2)
```

Out[66]:

	Member ID	Loan ID	Interest Rate	Amount Funded
0	149512	848058	19.05	7200
1	407046	659709	10.16	16000

In [67]:
```
#Code Block 55
df_subset = df[['Member ID', 'Loan ID','Interest Rate', 'Amount Funded']].copy()
df_subset.head()
```

Out[67]:

	Member ID	Loan ID	Interest Rate	Amount Funded
0	149512	848058	19.05	7200
1	407046	659709	10.16	16000
2	507531	601368	10.16	35000
3	513904	761341	6.03	21000
4	603349	885844	16.29	15000

In [68]:
```
#Code Block 56
df.iloc[:, [0]].head()
```

Out[68]:

	Member ID
0	149512
1	407046
2	507531
3	513904
4	603349

In [69]:
```
#Code Block 57
df.iloc[:, [1, 4, 2]].head()
```

Out[69]:

	Loan ID	Amount Funded	Origination Date
0	848058	7200	2019-08-18
1	659709	16000	2018-05-21
2	601368	35000	2018-01-01
3	761341	21000	2019-01-03
4	885844	15000	2019-11-17

In [70]:
```
#Code Block 58
df_smallset = df.iloc[0:10, [1, 4, 2]]
df_smallset
```

Out[70]:

	Loan ID	Amount Funded	Origination Date
0	848058	7200	2019-08-18
1	659709	16000	2018-05-21
2	601368	35000	2018-01-01
3	761341	21000	2019-01-03
4	885844	15000	2019-11-17
5	613337	1500	2018-01-16
6	789789	7500	2019-02-17
7	888522	35000	2019-11-20
8	746115	10000	2018-12-13
9	812348	3600	2019-04-24

In [71]:
```
#Code Block 59
df[df['Annual Income'] > 5000000]
```

Out[71]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Total Debt	Annual Income	Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employee Title
22304	3267055	726952	2018-11-14	13.11	14825	69548.0	7141778	11351.0	16000.0	36	B	Us postal service
29333	3979312	648723	2018-04-16	12.12	30000	329254.0	6100000	15219.0	16100.0	36	B	BOONE AND SONS JEWELERS

In [72]:
```
#Code Block 60
df[df['Annual Income'] > 5000000].index
```

Out[72]:
```
Int64Index([22304, 29333], dtype='int64')
```

In [73]:
```
#Code Block 61
df.iloc[[22304, 29333], [1, 4, 2]]
```

Out[73]:

	Loan ID	Amount Funded	Origination Date
22304	726952	14825	2018-11-14
29333	648723	30000	2018-04-16

In [74]:
```
#Code Block 62
varindex = df[df['Annual Income'] > 5000000].index
df.iloc[varindex, [1, 4, 2]]
```

Out[74]:

	Loan ID	Amount Funded	Origination Date
22304	726952	14825	2018-11-14
29333	648723	30000	2018-04-16

In [75]:
```
varindex
```

Out[75]:
```
Int64Index([22304, 29333], dtype='int64')
```

In [ ]:
```
#df.to_csv('data/Scenario5.csv')
```

In [ ]:
```

```