Scenario 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.Py01 - Install Anaconda**
  - https://www.anaconda.com/distribution/ (https://www.anaconda.com/distribution/)
- **C1.S2.Py02 - Intro to Anaconda**
- **C1.S2.Py03 - Intro to Jupyter Notebooks**
- **C1.S2.Py04 - Importing and Understanding Library Options**
  - numpy, pandas, pandas_profiling, scitkit learn, matplotlib, seaborn
- **C1.S2.Py05a - Import data as a DataFrame**
  - Read in a csv as a pandas dataframe
  - Read in an Excel as a apandas dataframe
- **C1.S2.Py05b - Understanding the Importing Data Options**
- **C1.S2.Py06 - How to Read Data**
  - .head(), .sample(), and .tail()
- **C1.S2.Py07 - How to Rename and Drop columns**
  - Rename columns, drop columns
- **C1.S2.Py08 - Understanding .info()**
- **C1.S2.Py09 - Understanding Data Types**

C1.S2.Py04 - Importing libraries and understanding the different Library Options

- Prior to executing any code in a Jupyter notebook, you need to import the libraries that you will use.

Three different thoughts on importing libraries

1. **ALL AT ONCE** Import all of the libraries that you will need first, that way you will only need to import once for the whole notebook.
2. **IMPORT AS NEEDED** - by importing when you need it, you do add unnecessary libraries to the that take up space.
3. **HYBRID IMPORTING** - Import the libraries that you know you will use, such as numpy and pandas, and then import other libraries when needed, such as sklearn.

Components of an import - When you import there are four aspects to consider:

1. **import** is used to import the entire library *(ex. import pandas)*.
2. **as** is used to give it a nickname or an easier name to use in your code *(ex. import pandas as pd)*. In the code you only need to type pd for it to recognize pandas.
3. **from** is used to import a part of the library but not all of it. This is important to do when the library is large, like sklearn (*ex. from sklearn.preprocessing import LabelEncoder*). By stating from, the code will go to sklearn and only import the LabelEncoder function. **NOTE: Always do this for sklearn**
4. **set_option** or **style** - allows you to set settings/styles or turn off/on settings to your liking. (*ex. pd.set_option('display.max_columns',500)* allows you to see a maximum of 500 columns at a time. If not set, it will cut off your columns at a low number.

   - Here is a guide for options for pandas - https://pandas.pydata.org/pandas-docs/stable/user_guide/options.html (https://pandas.pydata.org/pandas-docs/stable/user_guide/options.html)
   - Here is a guide to the different libraries that you may need for data science - https://dzone.com/articles/the-best-python-libraries-for-data-science-and-mac (https://dzone.com/articles/the-best-python-libraries-for-data-science-and-mac)

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',100) #allows for up to 100 columns to be displayed when
 viewing a dataframe

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

Different guides to styles galleries

- Style sheets reference — Matplotlib 3.1.1 documentation
  - https://matplotlib.org/3.1.1/gallery/style_sheets/style_sheets_reference.html (https://matplotlib.org/3.1.1/gallery/style_sheets/style_sheets_reference.html)
- Matplotlib Style Gallery - Tony S. Yu
  - https://tonysyu.github.io/raw_content/matplotlib-style-gallery/gallery.html (https://tonysyu.github.io/raw_content/matplotlib-style-gallery/gallery.html)

C1.S2.Py05a - Import Data as a DataFrame

C1.S2.Py05b - Understanding the Importing Data Options

https://pandas.pydata.org/pandas-docs/stable/api.html#flat-file (https://pandas.pydata.org/pandas-docs/stable/api.html#flat-file)

Four different ways to import data

1. **Import csv** - local csv files are universally the easiest files to import and are usually the fastest files to import as well.
2. **Import Excel file** - allows you to import multiple sheets from the same file. This is usually a slower process.
3. **Import files from URL** - if the files are not local then you can point to a URL to import.
4. **Import a text files** - this is similar to a csv but you need to identify the separator, such as tab or space.

Import a csv file

- **header** specifies that the top row is the label for the column - set it *=0* for the first row or *=None* if there is no label in the first row.
- **column_index** specifies that the first column is the index, which is a unique identifier. set it *=0* for the first column or *=None* if you prefer to leave the first column as a regular column. *If the column is normal column set it =None.
- **encoding** sometimes there is an issue reading in the data because it cannot detemrine the correct encoding. When this occurs, set ***encoding* = '*utf8*'**.
  Ex. *df = pd.read_csv('data/Appleton.csv', encoding = 'utf8')*

In [2]:
```
#Code Block 2
df_column = pd.read_csv('data/LoanAnalysis_RawData.csv', index_col = 0, header=0)
    #sets the first column to the index
    # and the top row as the headers
df = pd.read_csv('data/LoanAnalysis_RawData.csv', index_col = None, header=0)
    #DOES NOT set the first column to the index
    # and the top row as the headers
```

In [3]:
```
#Code Block 3
df_column.head(2)
```

Out[3]:

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

In [4]:
```
#Code Block 4
df.head(2)
```

Out[4]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Borrower Total Debt	Annual Income	Balance on 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	

Import an Excel file

- When importing a xlsx file, it will bring in the entire sheet. You can then take individual sheets and create dataframes using ***parse***.

In [5]:
```
#Code Block 5
df_excel =  pd.ExcelFile('data/Loan Analysis - Raw Data.xlsx')
print(df_excel.sheet_names)
```

```
['Loan Subset', 'Sheet2']
```

In [6]:
```
#Code Block 6
df_sheet1 = df_excel.parse('Loan Subset')
df_sheet1.head(2)
```

Out[6]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Borrower Total Debt	Annual Income	Balance on Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employee Title	Le Empl
0	1524478	600947	2018-01-01	7.90	5875	28154	29643.0	4405	6600	36	A	Prout Levangie	
1	1682817	600960	2018-01-01	6.03	10800	15175	86000.0	8030	22400	36	A	SmartPros Ltd.	

In [7]:
```
#Code Block 7
df_sheet1_1 = df_excel.parse(0)
df_sheet1_1.head(2)
```

Out[7]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Borrower Total Debt	Annual Income	Balance on Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employee Title	Le Empl
0	1524478	600947	2018-01-01	7.90	5875	28154	29643.0	4405	6600	36	A	Prout Levangie	
1	1682817	600960	2018-01-01	6.03	10800	15175	86000.0	8030	22400	36	A	SmartPros Ltd.	

In [8]:
```
#Code Block 8
df_sheet2 = df_excel.parse('Sheet2')
df_sheet2.head(2)
```

Out[8]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Borrower Total Debt	Annual Income	Balance on Revolving Accounts	Total Revolving Credit Line	Term	Grade
0	1805383	601168	2018-01-01	20.49	35000	292774	110000	31688	42900	60	E
1	1807246	601187	2018-01-01	17.27	10500	266580	55000	8256	16700	60	C

In [9]:
```
#Code Block 9
df_sheet2_1 = df_excel.parse(1)
df_sheet2_1.head(2)
```

Out[9]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Borrower Total Debt	Annual Income	Balance on Revolving Accounts	Total Revolving Credit Line	Term	Grade
0	1805383	601168	2018-01-01	20.49	35000	292774	110000	31688	42900	60	E
1	1807246	601187	2018-01-01	17.27	10500	266580	55000	8256	16700	60	C

Import URL

NOTE: When importing from specific repositories, make sure to follow their path requirements. Github needs *raw.githubusercontent.com* then the path.

In [10]:
```
#Code Block 10
url = 'https://raw.githubusercontent.com/capigian/PythonWorkshop/master/AppletonOriginal.csv'
#url is a variable that is created that is pointed to the ile online.

df_Appleton_url = pd.read_csv(url, header = 0, index_col=None)
df_Appleton_url.head()
```

Out[10]:

	member_id	loan_amnt	orig_date	term	int_rate	installment	risk_factor	annual_inc	delinq_2yrs	inq_last_6mths	mths_s
0	3411415	2000	12/24/16	36	17.27	71.58	-3	26000.0	0	1	
1	3410838	7750	12/24/16	36	13.11	261.54	-2	39500.0	1	2	
2	3176905	4500	12/24/16	36	19.05	165.07	-4	55000.0	0	0	
3	3420387	20850	12/24/16	60	17.77	526.85	-4	143784.0	0	0	
4	3420200	12000	12/24/16	36	14.33	412.06	-2	44000.0	2	1	

Read in specific columns from a file

- sometimes you only want to read in a few columns, to do this you can use ***usecols***

In [11]:
```
#Code Block 11
df_4 = pd.read_csv("data/LoanAnalysis_RawData.csv", usecols = ['Member ID','Loan ID', 'Origination Date', 'Interest Rate'])
df_4.head()
```

Out[11]:

	Member ID	Loan ID	Origination Date	Interest Rate
0	149512	848058	8/18/19	19.05
1	407046	659709	5/21/18	10.16
2	507531	601368	1/1/18	10.16
3	513904	761341	1/3/19	6.03
4	603349	885844	11/17/19	16.29

C1.S2.Py06 - How to Read Data

- **.head( )** - shows the first 5 records (default). If you add a number inside of the parentheses, then that is how many records will be shown. (*ex. .head(15) will show the top 15 records.
- **.tail( )** - shows the last 5 records. (*ex. .tail(10) will show the bottom 10 records.
- **.sample ( )** - shows 1 random record. If you include a number in the parentheses then it will show a random number for that number. (*ex. .sample(10) - will show 10 random records.)
- **.info( )** - shows every column with the data type and the total number of records. Shown in next video.

In [12]:
```
#Code Block 12
df.head()
```

Out[12]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Borrower Total Debt	Annual Income	Balance on Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employee Title	Em
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	
2	507531	601368	1/1/18	10.16	35000	60019.0	130000	23025.0	55800.0	36	B	gSEMI	
3	513904	761341	1/3/19	6.03	21000	37603.0	120000	18641.0	85031.0	36	A	Fidelity Investments	
4	603349	885844	11/17/19	16.29	15000	227890.0	72000	11702.0	26300.0	36	C	NaN	

In [13]:
```
#Code Block 13
df_column.head()
```

Out[13]:

Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Borrower Total Debt	Annual Income	Balance on Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employee Title	Len Emplo
149512	848058	8/18/19	19.05	7200	154930.0	58000	3874.0	4300.0	36	D	Arkwright	
407046	659709	5/21/18	10.16	16000	29116.0	55000	6840.0	24800.0	36	B	School	
507531	601368	1/1/18	10.16	35000	60019.0	130000	23025.0	55800.0	36	B	gSEMI	
513904	761341	1/3/19	6.03	21000	37603.0	120000	18641.0	85031.0	36	A	Fidelity Investments	
603349	885844	11/17/19	16.29	15000	227890.0	72000	11702.0	26300.0	36	C	NaN	

In [14]:
```
#Code Block 14
df.head(10)
```

Out[14]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Borrower Total Debt	Annual Income	Balance on Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employee Title	Em
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	
2	507531	601368	1/1/18	10.16	35000	60019.0	130000	23025.0	55800.0	36	B	gSEMI	
3	513904	761341	1/3/19	6.03	21000	37603.0	120000	18641.0	85031.0	36	A	Fidelity Investments	
4	603349	885844	11/17/19	16.29	15000	227890.0	72000	11702.0	26300.0	36	C	NaN	
5	656281	613337	1/16/18	14.33	1500	11451.0	75000	3362.0	3700.0	36	C	Select Therapies	
6	735990	789789	2/17/19	7.62	7500	265809.0	92000	6419.0	43000.0	36	A	TD Bank	
7	771211	888522	11/20/19	21.49	35000	354982.0	114000	38651.0	79800.0	60	E	Nevada Gaming Control Board	
8	778284	746115	12/13/18	6.03	10000	152402.0	108000	4653.0	46100.0	36	A	FlightStats, Inc.	
9	780866	812348	4/24/19	11.14	3600	175788.0	65000	12936.0	39400.0	36	B	City of Ithaca	

In [15]:
```
#Code Block 15
df.tail()
```

Out[15]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Borrower Total Debt	Annual Income	Balance on Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employee Title
30066	4068778	908730	12/18/19	6.62	17000	203808.0	95000	16801.0	40300.0	36	A	Traverse City Area Public School
30067	4068801	688645	8/20/18	13.11	14400	58904.0	81000	32651.0	40200.0	36	B	Science, Management & Resources
30068	4068843	657946	5/13/18	7.90	16000	372771.0	110000	23691.0	31500.0	36	A	Bristol Hospital
30069	4068857	906205	12/15/19	6.62	11200	187717.0	108000	37822.0	66400.0	36	A	Nokia Siemens Network
30070	4076727	630530	2/14/18	11.14	8000	19052.0	35000	6602.0	10600.0	36	B	pa liquor control board

In [16]:
```
#Code Block 16
df.tail(7)
```

Out[16]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Borrower Total Debt	Annual Income	Balance on Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employee Title
30064	4068726	640807	4/1/18	11.14	8975	95440.0	41000	7849.0	14700.0	36	B	Non-profit
30065	4068756	661996	5/29/18	14.09	8400	339768.0	73000	9249.0	13100.0	36	B	State of Oregon
30066	4068778	908730	12/18/19	6.62	17000	203808.0	95000	16801.0	40300.0	36	A	Traverse City Area Public School
30067	4068801	688645	8/20/18	13.11	14400	58904.0	81000	32651.0	40200.0	36	B	Science, Management & Resources
30068	4068843	657946	5/13/18	7.90	16000	372771.0	110000	23691.0	31500.0	36	A	Bristol Hospital
30069	4068857	906205	12/15/19	6.62	11200	187717.0	108000	37822.0	66400.0	36	A	Nokia Siemens Network
30070	4076727	630530	2/14/18	11.14	8000	19052.0	35000	6602.0	10600.0	36	B	pa liquor control board

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

Out[17]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Borrower Total Debt	Annual Income	Balance on Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employee Title
11472	1899320	703065	10/3/18	15.31	15000	241902.0	65000	13983.0	35300.0	36	C	SRMC

In [19]:
```
#Code Block 18
df.sample(5)
```

Out[19]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Borrower Total Debt	Annual Income	Balance on Revolving Accounts	Total Revolving Credit Line	Term	Grade	Employee Title
22614	3409367	913091	12/25/19	15.80	7000	3477.0	136000	3477.0	36700.0	36	C	Alta Via Consulting
21070	2917679	699847	9/27/18	17.77	12000	307275.0	74000	3476.0	19200.0	36	D	California Department of Correctior
11063	1893976	690468	8/29/18	14.33	9550	124088.0	53000	6077.0	9300.0	36	C	MV Commercia Construction LLC
6023	1815622	894003	12/1/19	15.31	6000	33688.0	57000	24715.0	31100.0	36	C	ADF
13142	1973104	677172	7/15/18	16.29	6000	34231.0	68000	5748.0	9100.0	36	C	Bank o America

C1.S2.Py07 - How to Rename and Drop Columns

When importing data, it is a good idea to make sure that names for columns and concise and descriptive. Also, copying dataframes is a good idea to ensure original data.

Creating a copy of a dataframe

- Prior to making changes with your DataFrame, create a copy.
- .copy()
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.copy.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.copy.html)

In [20]:
```
#Code Block 19
df.head(2)
```

Out[20]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Borrower Total Debt	Annual Income	Balance on 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 [21]:
```
#Code Block 20
df_copy = df.copy()
df_copy.head(2)
```

Out[21]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Borrower Total Debt	Annual Income	Balance on 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	

Create a subset of columns for a new DataFrame

In [22]:
```
#Code Block 21
df_customer = df[['Member ID','Loan ID','Origination Date','Interest Rate','Amount Funded',
'Borrower Total Debt','Annual Income','Employee Title','Length of Employment','Home Ownershi
p']].copy()
df_customer.head()
```

Out[22]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Borrower Total Debt	Annual Income	Employee Title	Length of Employment	Home Ownership
0	149512	848058	8/18/19	19.05	7200	154930.0	58000	Arkwright	9.0	RENT
1	407046	659709	5/21/18	10.16	16000	29116.0	55000	School	4.0	RENT
2	507531	601368	1/1/18	10.16	35000	60019.0	130000	gSEMI	8.0	RENT
3	513904	761341	1/3/19	6.03	21000	37603.0	120000	Fidelity Investments	10.0	RENT
4	603349	885844	11/17/19	16.29	15000	227890.0	72000	NaN	NaN	MORTGAGE

Rename columns in a dataframe

- Set Dataframe to the rename function
- Example: df=df.rename(columns = {'originalcolumn':'newcolumn'})
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.rename.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.rename.html)

In [23]:
```
#Code Block 22
df.head(2)
```

Out[23]:

	Member ID	Loan ID	Origination Date	Interest Rate	Amount Funded	Borrower Total Debt	Annual Income	Balance on 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 [24]:
```
#Code Block 23
df=df.rename(columns = {'Borrower Total Debt':'Total Debt', \
                        'Balance on Revolving Accounts':'Revolving Accounts'})
df.head(2)
```

Out[24]:

	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 [25]:
```
#Code Block 24
df = df.drop('Notes', axis = 1)
df.head(2)
```

Out[25]:

	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 [ ]:
```
#df.to_csv('data/Scenario4.csv')
```