DATA CLEANSING THE LENDING CLUB LOAN DATASET

WITH PYTHON

DR. ALVIN ANG

CONTENTS

I.

STEP 1: READING IN THE DATA

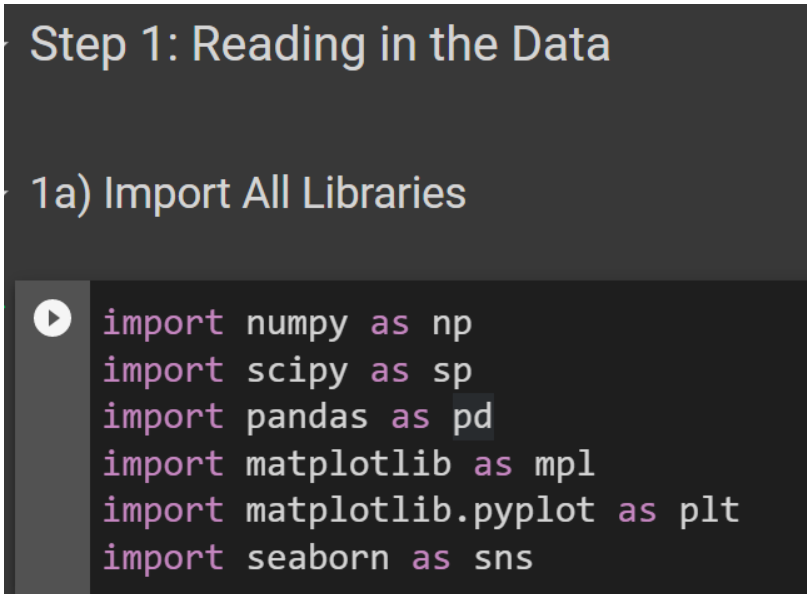
IPYNB:

- https://www.alvinang.sg/s/Data_Cleansing_the_Lending_Club_Loan_Dataset_by_Dr_Alvin_Ang.ipynb

FILES:

- https://www.alvinang.sg/s/LendingClubLoan200-rows.csv
- https://www.alvinang.sg/s/LCDataDictionary.xlsx

A. IMPORT ALL LIBRARIES

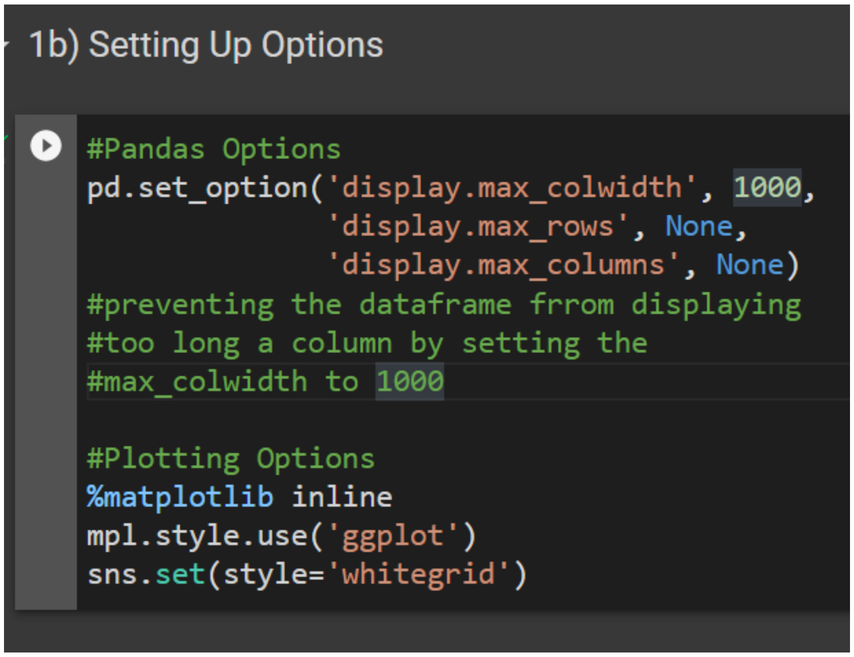


Step 1: Reading in the Data

1a) Import All Libraries

```
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
```

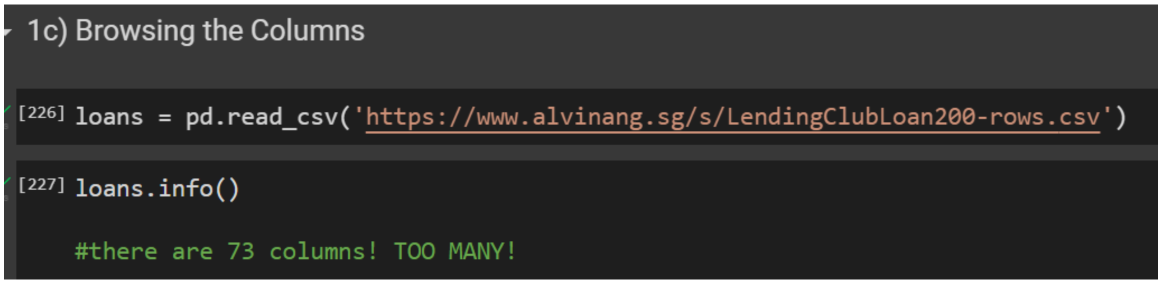
B. SETTING UP OPTIONS



1b) Setting Up Options

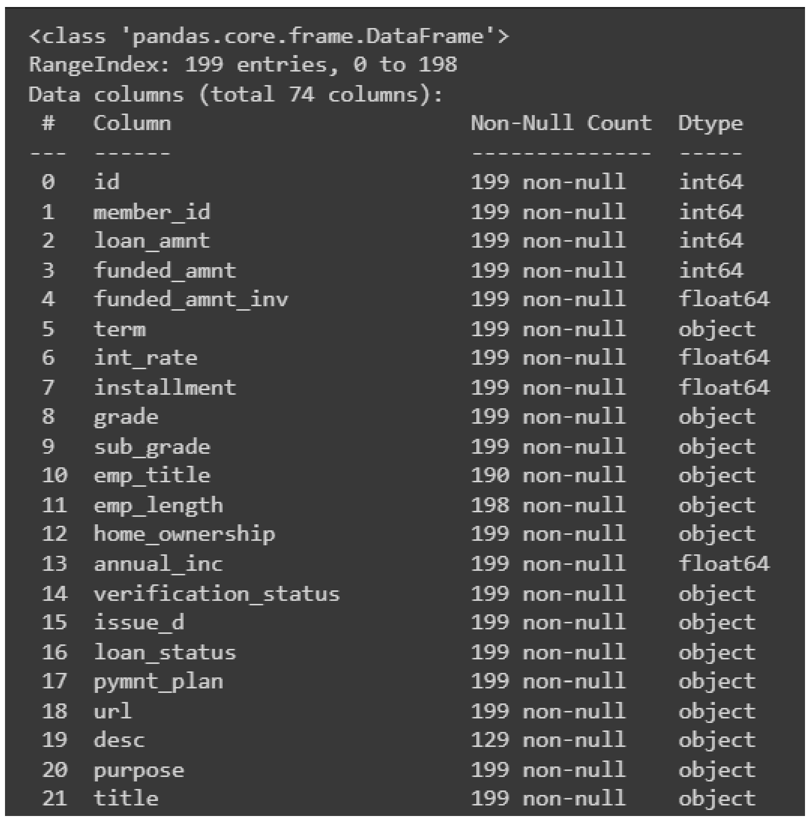
```
#Pandas Options
pd.set_option('display.max_colwidth', 1000,
              'display.max_rows', None,
              'display.max_columns', None)
#preventing the dataframe frrom displaying
#too long a column by setting the
#max_colwidth to 1000

#Plotting Options
%matplotlib inline
mpl.style.use('ggplot')
sns.set(style='whitegrid')
```

C. PEEKING AT THE DATA

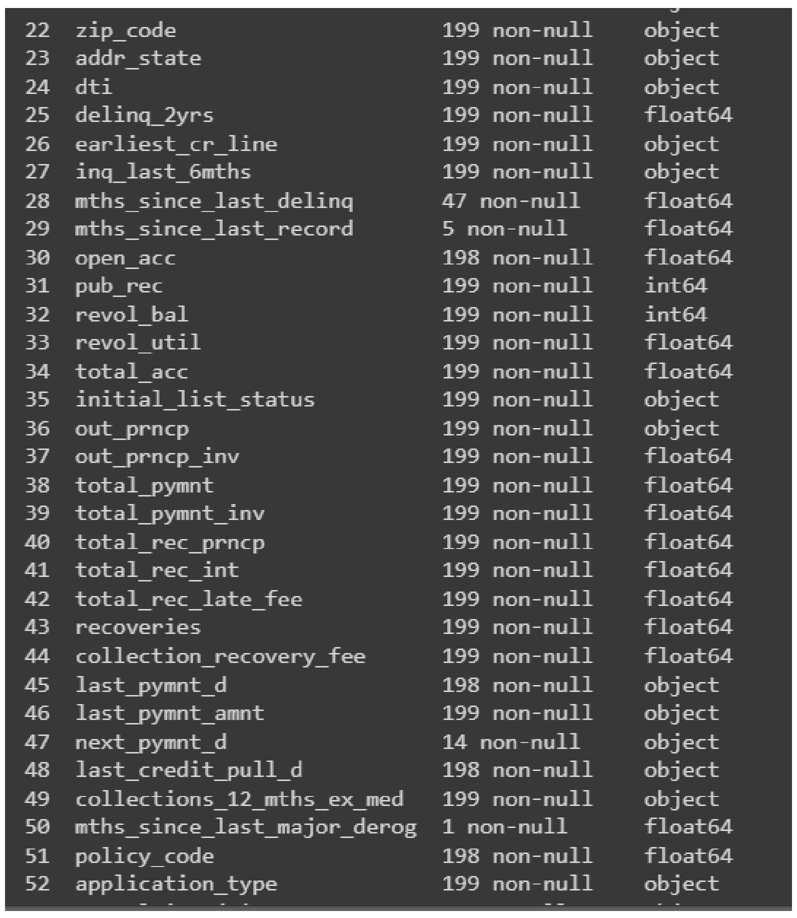


1c) Browsing the Columns

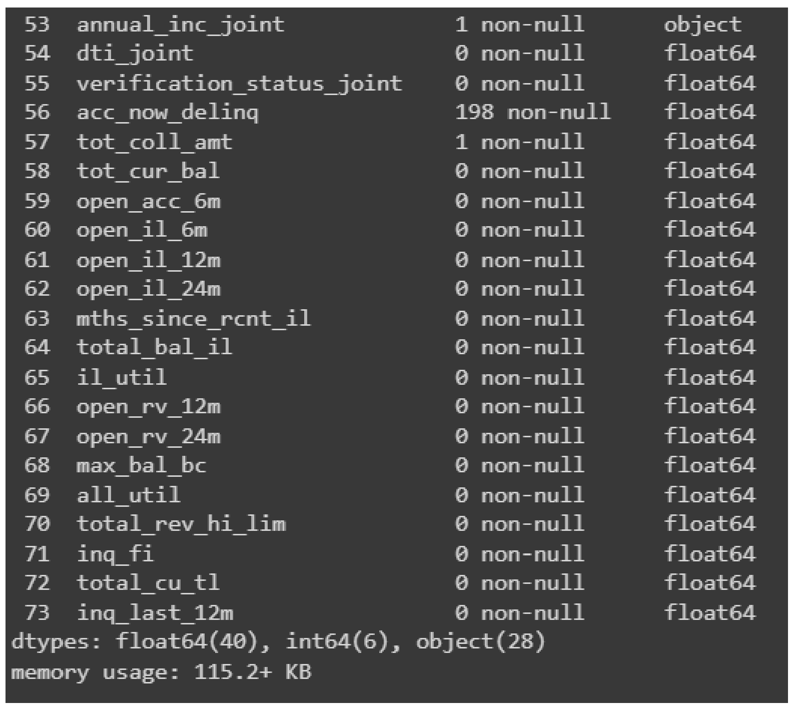
```
[226] loans = pd.read_csv('https://www.alvinang.sg/s/LendingClubLoan200-rows.csv')

[227] loans.info()

      #there are 73 columns! TOO MANY!
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 199 entries, 0 to 198
Data columns (total 74 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   id                           199 non-null    int64
 1   member_id                    199 non-null    int64
 2   loan_amnt                    199 non-null    int64
 3   funded_amnt                  199 non-null    int64
 4   funded_amnt_inv              199 non-null    float64
 5   term                         199 non-null    object
 6   int_rate                     199 non-null    float64
 7   installment                  199 non-null    float64
 8   grade                        199 non-null    object
 9   sub_grade                    199 non-null    object
 10  emp_title                    190 non-null    object
 11  emp_length                   198 non-null    object
 12  home_ownership               199 non-null    object
 13  annual_inc                   199 non-null    float64
 14  verification_status          199 non-null    object
 15  issue_d                      199 non-null    object
 16  loan_status                  199 non-null    object
 17  pymnt_plan                   199 non-null    object
 18  url                          199 non-null    object
 19  desc                         129 non-null    object
 20  purpose                      199 non-null    object
 21  title                        199 non-null    object
```

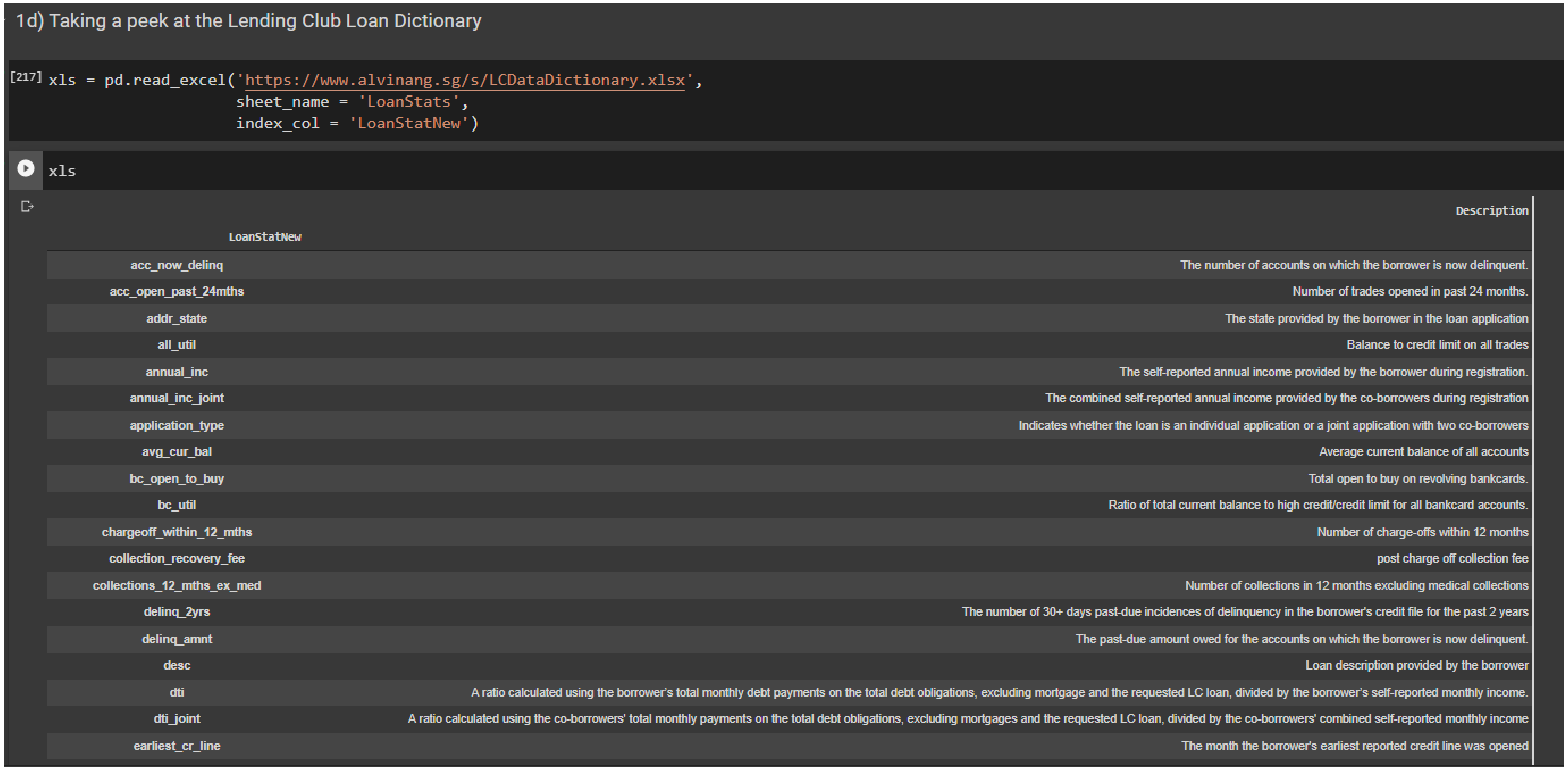


```
 22  zip_code                     199 non-null    object
 23  addr_state                   199 non-null    object
 24  dti                          199 non-null    object
 25  delinq_2yrs                  199 non-null    float64
 26  earliest_cr_line             199 non-null    object
 27  inq_last_6mths               199 non-null    object
 28  mths_since_last_delinq       47 non-null     float64
 29  mths_since_last_record       5 non-null      float64
 30  open_acc                     198 non-null    float64
 31  pub_rec                      199 non-null    int64
 32  revol_bal                    199 non-null    int64
 33  revol_util                   199 non-null    float64
 34  total_acc                    199 non-null    float64
 35  initial_list_status          199 non-null    object
 36  out_prncp                    199 non-null    object
 37  out_prncp_inv                199 non-null    float64
 38  total_pymnt                  199 non-null    float64
 39  total_pymnt_inv              199 non-null    float64
 40  total_rec_prncp              199 non-null    float64
 41  total_rec_int                199 non-null    float64
 42  total_rec_late_fee           199 non-null    float64
 43  recoveries                   199 non-null    float64
 44  collection_recovery_fee      199 non-null    float64
 45  last_pymnt_d                 198 non-null    object
 46  last_pymnt_amnt              199 non-null    object
 47  next_pymnt_d                 14 non-null     object
 48  last_credit_pull_d           198 non-null    object
 49  collections_12_mths_ex_med   199 non-null    object
 50  mths_since_last_major_derog  1 non-null      float64
 51  policy_code                  198 non-null    float64
 52  application_type             199 non-null    object
```



```
 53  annual_inc_joint             1 non-null      object
 54  dti_joint                    0 non-null      float64
 55  verification_status_joint    0 non-null      float64
 56  acc_now_delinq               198 non-null    float64
 57  tot_coll_amt                 1 non-null      float64
 58  tot_cur_bal                  0 non-null      float64
 59  open_acc_6m                  0 non-null      float64
 60  open_il_6m                   0 non-null      float64
 61  open_il_12m                  0 non-null      float64
 62  open_il_24m                  0 non-null      float64
 63  mths_since_rcnt_il           0 non-null      float64
 64  total_bal_il                 0 non-null      float64
 65  il_util                      0 non-null      float64
 66  open_rv_12m                  0 non-null      float64
 67  open_rv_24m                  0 non-null      float64
 68  max_bal_bc                   0 non-null      float64
 69  all_util                     0 non-null      float64
 70  total_rev_hi_lim             0 non-null      float64
 71  inq_fi                       0 non-null      float64
 72  total_cu_tl                  0 non-null      float64
 73  inq_last_12m                 0 non-null      float64
dtypes: float64(40), int64(6), object(28)
memory usage: 115.2+ KB
```

D. TAKING A PEEK AT THE LENDING CLIUB LOAN DICTIONARY



1d) Taking a peek at the Lending Club Loan Dictionary

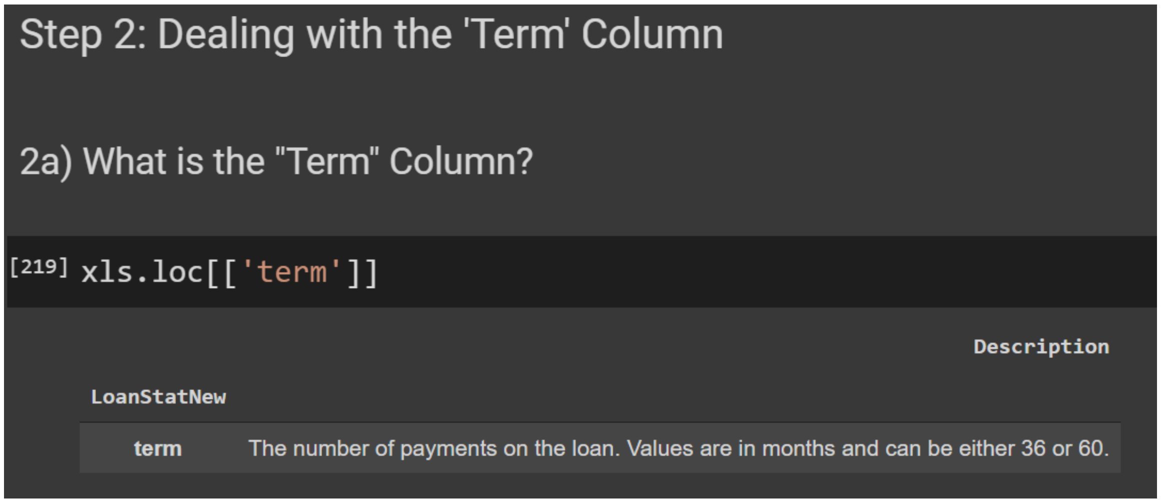
```
[217] xls = pd.read_excel('https://www.alvinang.sg/s/LCDataDictionary.xlsx',
                    sheet_name = 'LoanStats',
                    index_col = 'LoanStatNew')
```

```
xls
```

	Description
LoanStatNew	
acc_now_delinq	The number of accounts on which the borrower is now delinquent.
acc_open_past_24mths	Number of trades opened in past 24 months.
addr_state	The state provided by the borrower in the loan application
all_util	Balance to credit limit on all trades
annual_inc	The self-reported annual income provided by the borrower during registration.
annual_inc_joint	The combined self-reported annual income provided by the co-borrowers during registration
application_type	Indicates whether the loan is an individual application or a joint application with two co-borrowers
avg_cur_bal	Average current balance of all accounts
bc_open_to_buy	Total open to buy on revolving bankcards.
bc_util	Ratio of total current balance to high credit/credit limit for all bankcard accounts.
chargeoff_within_12_mths	Number of charge-offs within 12 months
collection_recovery_fee	post charge off collection fee
collections_12_mths_ex_med	Number of collections in 12 months excluding medical collections
delinq_2yrs	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years
delinq_amnt	The past-due amount owed for the accounts on which the borrower is now delinquent.
desc	Loan description provided by the borrower
dti	A ratio calculated using the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower's self-reported monthly income.
dti_joint	A ratio calculated using the co-borrowers' total monthly payments on the total debt obligations, excluding mortgages and the requested LC loan, divided by the co-borrowers' combined self-reported monthly income
earliest_cr_line	The month the borrower's earliest reported credit line was opened

II. STEP 2: DEALING WITH THE “TERM” COLUMN

A. WHAT IS THE “TERM” COLUMN?



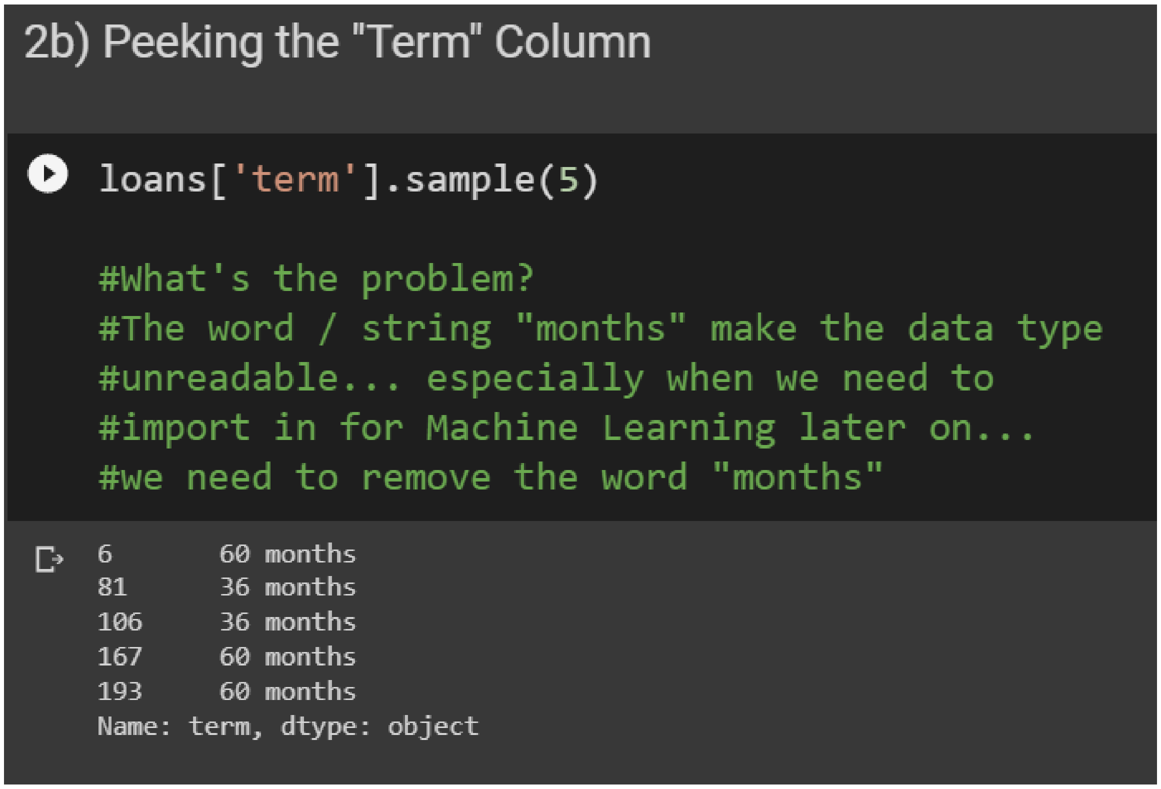
Step 2: Dealing with the 'Term' Column

2a) What is the "Term" Column?

```
[219] xls.loc[['term']]
```

LoanStatNew	Description
term	The number of payments on the loan. Values are in months and can be either 36 or 60.

B. PEEKING THE “TERM” COLUMN

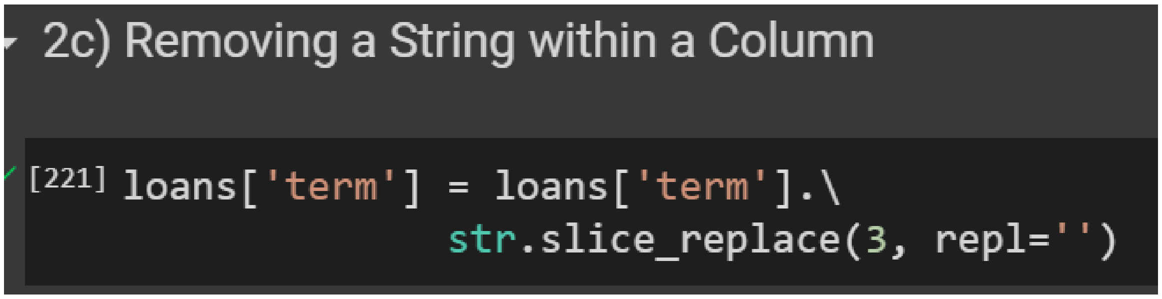


2b) Peeking the "Term" Column

```
loans['term'].sample(5)

#What's the problem?
#The word / string "months" make the data type
#unreadable... especially when we need to
#import in for Machine Learning later on...
#we need to remove the word "months"
```

```
6        60 months
81       36 months
106      36 months
167      60 months
193      60 months
Name: term, dtype: object
```

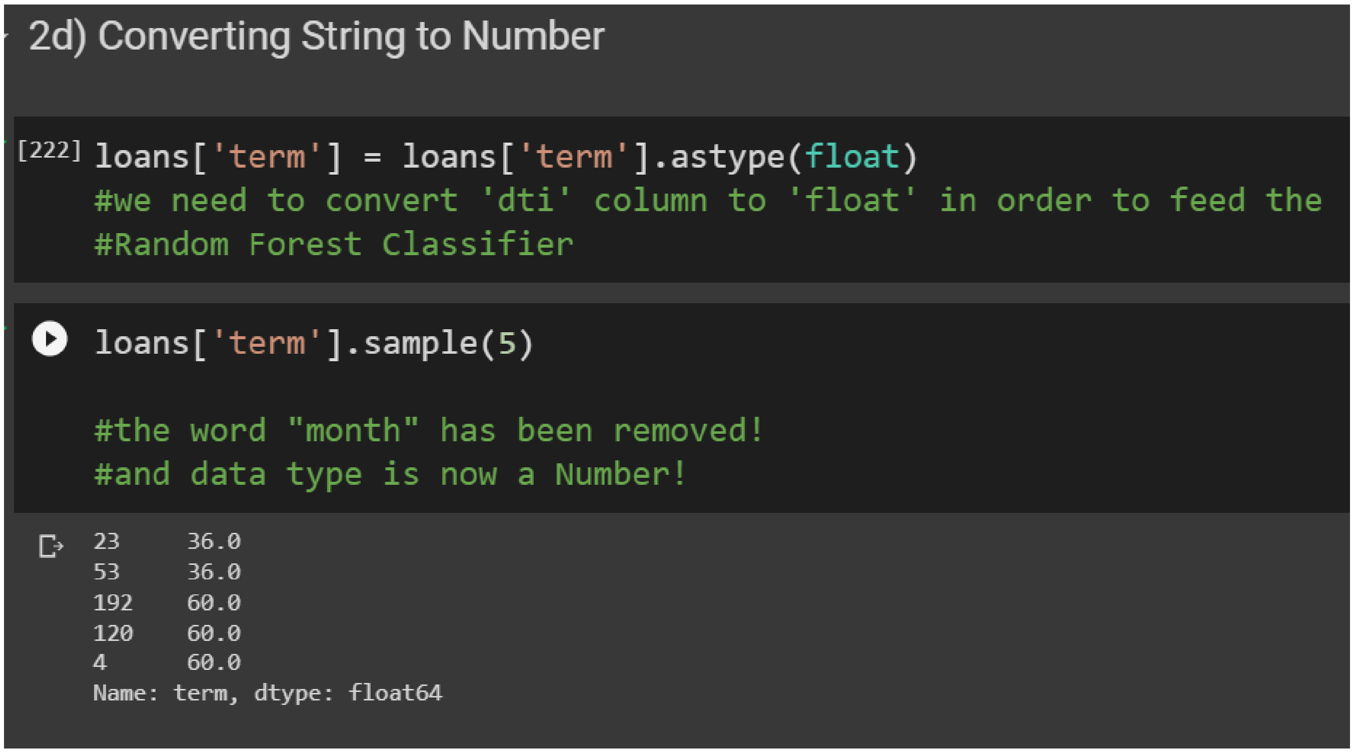
C. REMOVING A STRING WITHIN A COLUMN



2c) Removing a String within a Column

```
[221] loans['term'] = loans['term'].\
                      str.slice_replace(3, repl='')
```

D. CONVERTING STRING TO NUMBER



2d) Converting String to Number

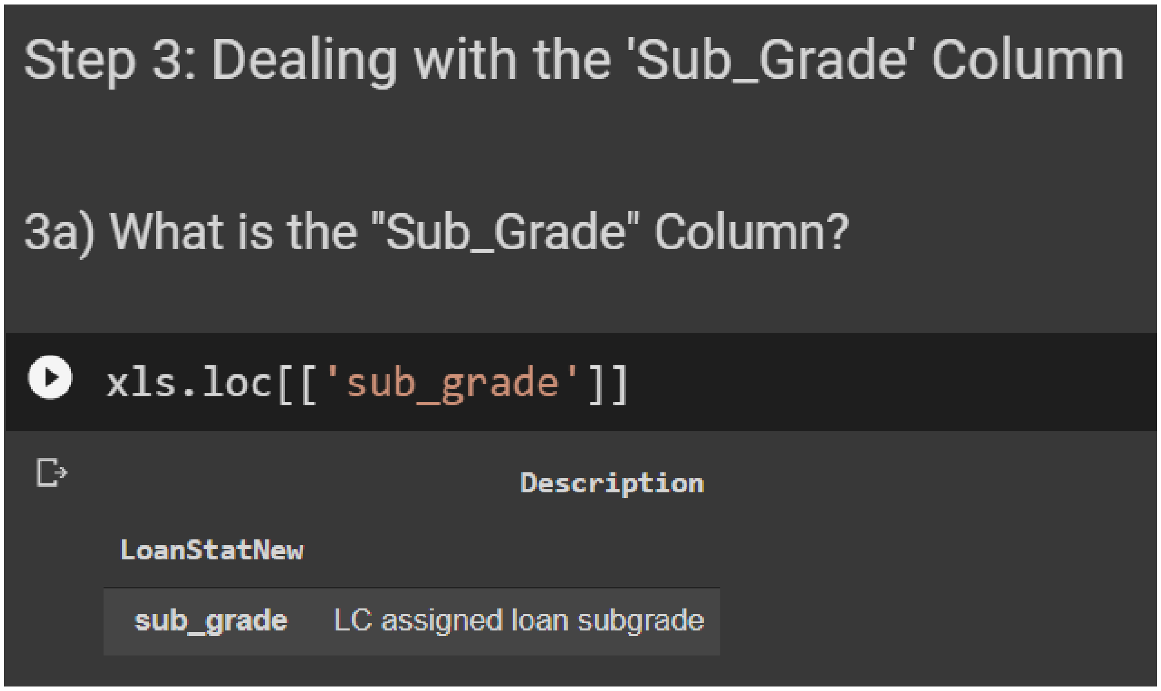
```
[222] loans['term'] = loans['term'].astype(float)
      #we need to convert 'dti' column to 'float' in order to feed the
      #Random Forest Classifier
```

```
loans['term'].sample(5)

#the word "month" has been removed!
#and data type is now a Number!
```

```
23     36.0
53     36.0
192    60.0
120    60.0
4      60.0
Name: term, dtype: float64
```

III. STEP 3: DEALING WITH THE "SUB_GRADE" COLUMN

A. WHAT IS THE "SUB_GRADE" COLUMN?



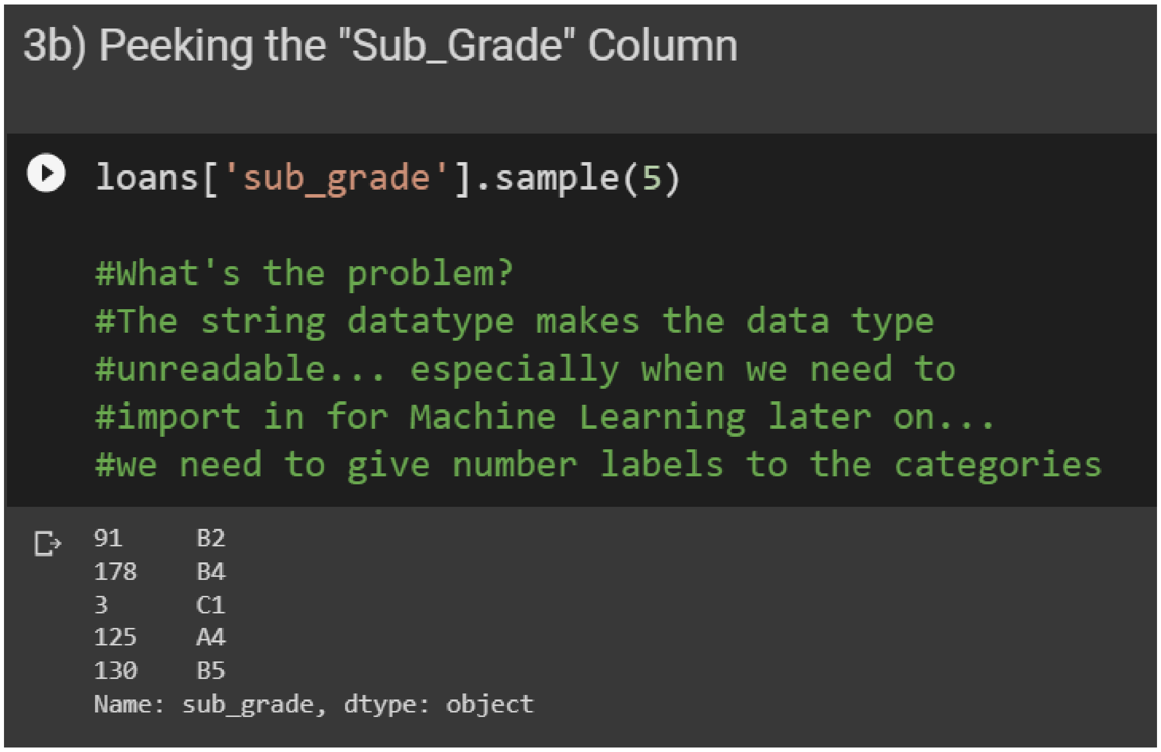
Step 3: Dealing with the 'Sub_Grade' Column

3a) What is the "Sub_Grade" Column?

```
xls.loc[['sub_grade']]
```

```
                          Description
LoanStatNew
sub_grade    LC assigned loan subgrade
```

B. PEEKING THE "SUB_GRADE" COLUMN

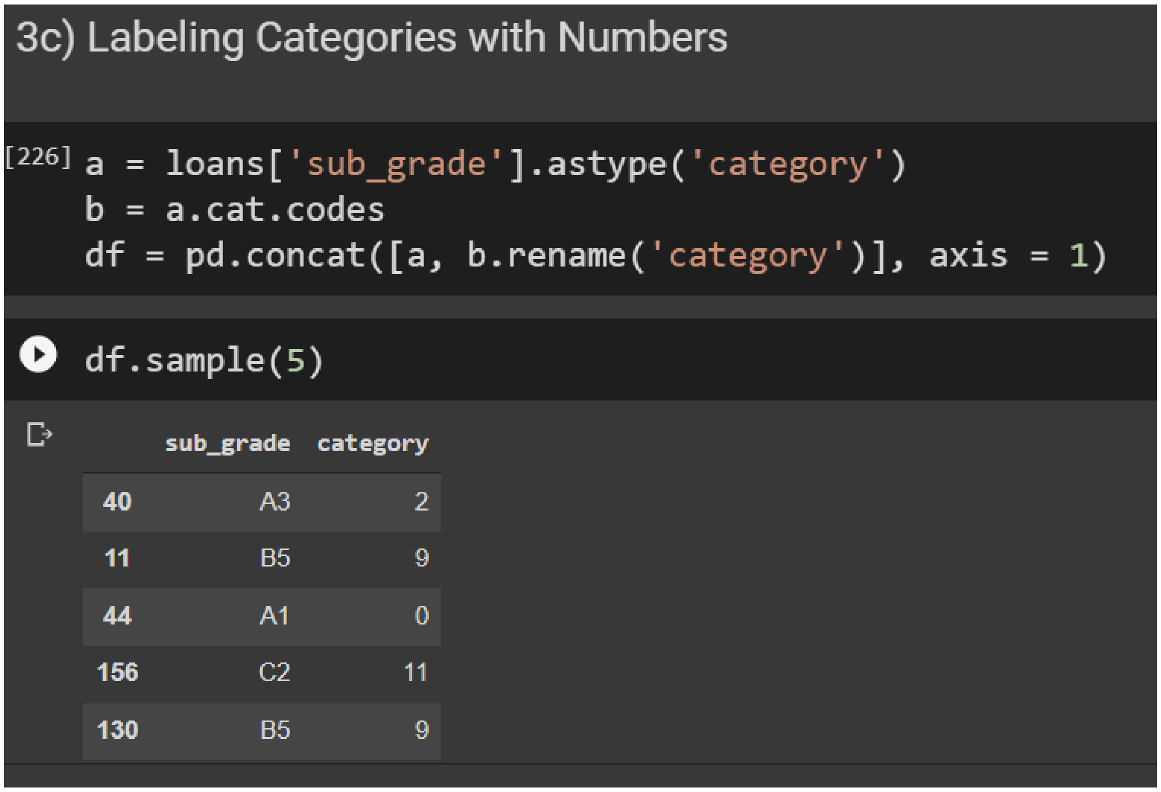


3b) Peeking the "Sub_Grade" Column

```
loans['sub_grade'].sample(5)

#What's the problem?
#The string datatype makes the data type
#unreadable... especially when we need to
#import in for Machine Learning later on...
#we need to give number labels to the categories
```

```
91     B2
178    B4
3      C1
125    A4
130    B5
Name: sub_grade, dtype: object
```

C. LABELING CATEGORIES WITH NUMBERS

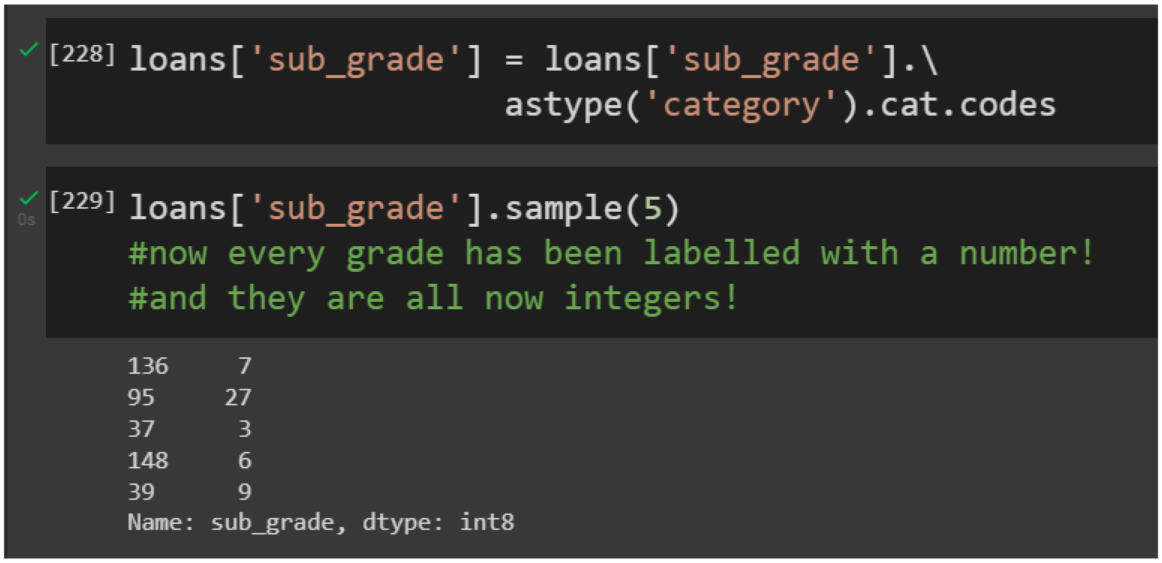


3c) Labeling Categories with Numbers

```
[226] a = loans['sub_grade'].astype('category')
      b = a.cat.codes
      df = pd.concat([a, b.rename('category')], axis = 1)
```

```
df.sample(5)
```

	sub_grade	category
40	A3	2
11	B5	9
44	A1	0
156	C2	11
130	B5	9

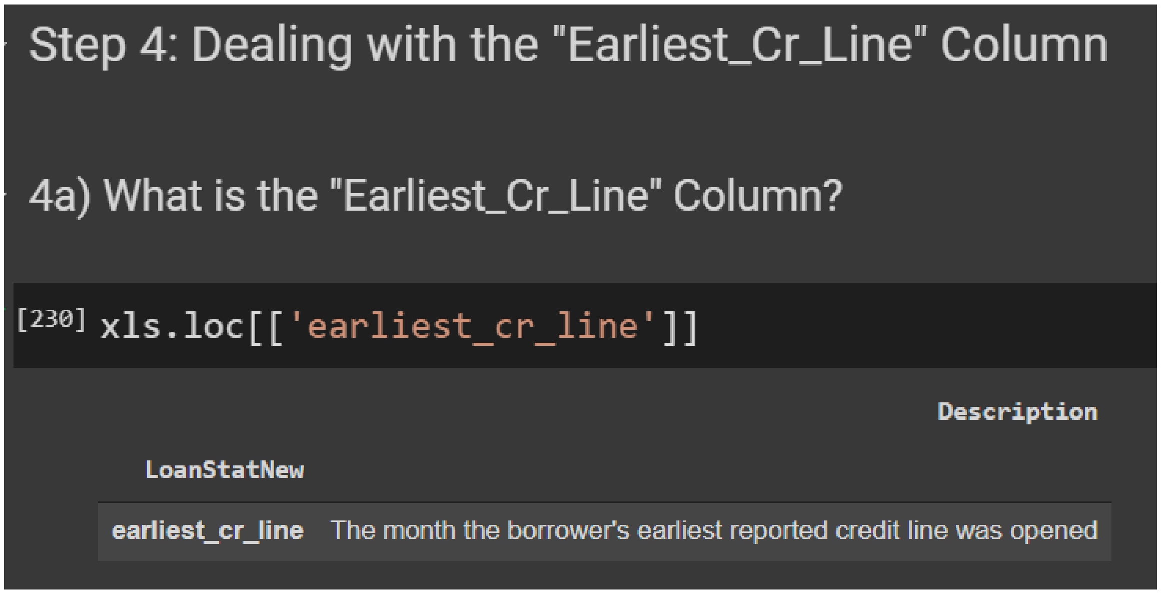


```
[228] loans['sub_grade'] = loans['sub_grade'].\
                          astype('category').cat.codes
```

```
[229] loans['sub_grade'].sample(5)
      #now every grade has been labelled with a number!
      #and they are all now integers!

      136     7
      95     27
      37      3
      148     6
      39      9
      Name: sub_grade, dtype: int8
```

IV. STEP 4: DEALING WITH THE "EARLIEST_CR_LINE" COLUMN

A. WHAT IS THE "EARLIEST_CR_LINE" COLUMN?



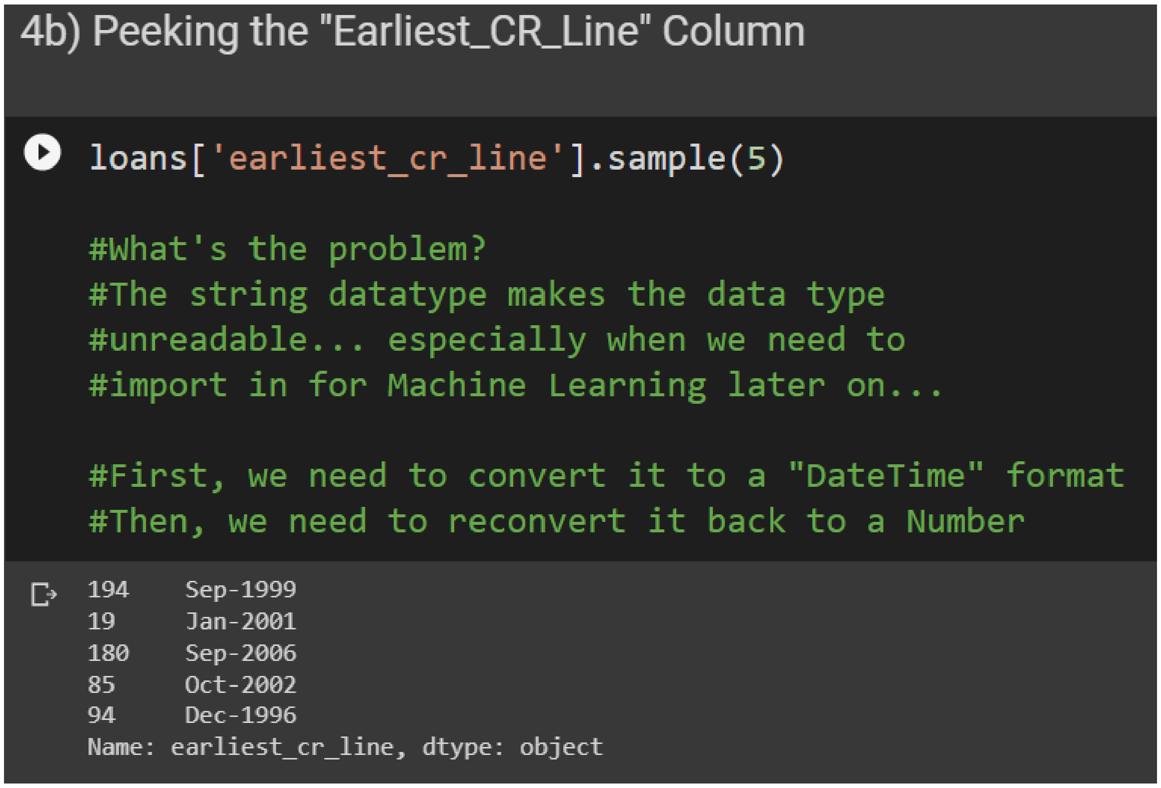
Step 4: Dealing with the "Earliest_Cr_Line" Column

4a) What is the "Earliest_Cr_Line" Column?

```
[230] xls.loc[['earliest_cr_line']]
```

LoanStatNew	Description
earliest_cr_line	The month the borrower's earliest reported credit line was opened

B. PEEKING THE "EARLIEST_CR_LINE" COLUMN

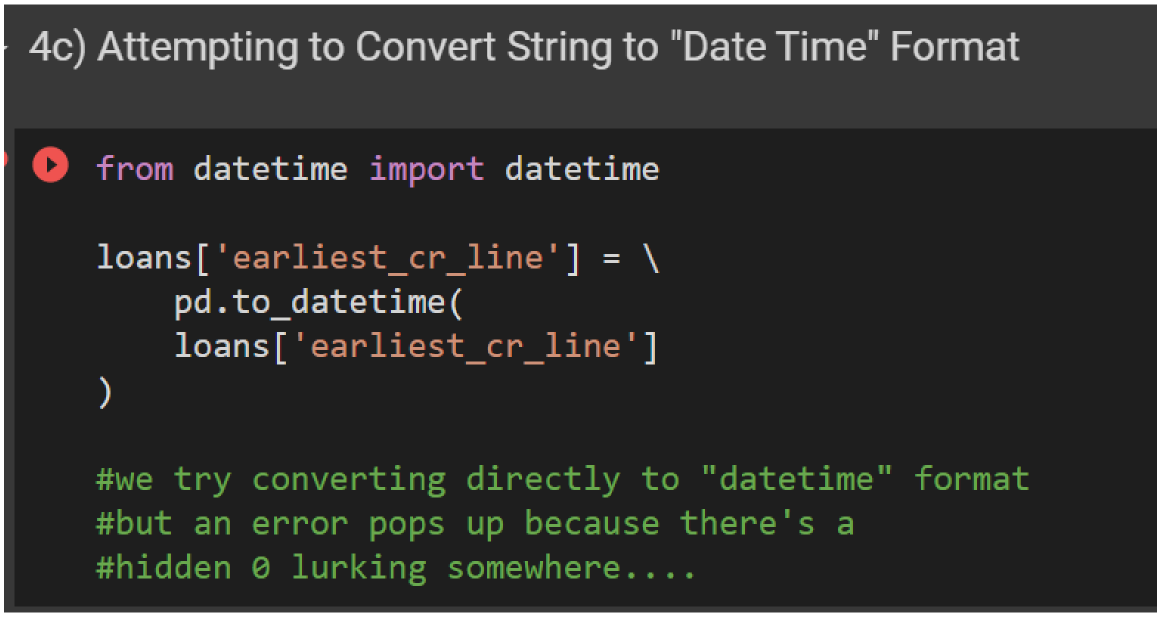


4b) Peeking the "Earliest_CR_Line" Column

```
loans['earliest_cr_line'].sample(5)

#What's the problem?
#The string datatype makes the data type
#unreadable... especially when we need to
#import in for Machine Learning later on...

#First, we need to convert it to a "DateTime" format
#Then, we need to reconvert it back to a Number
```

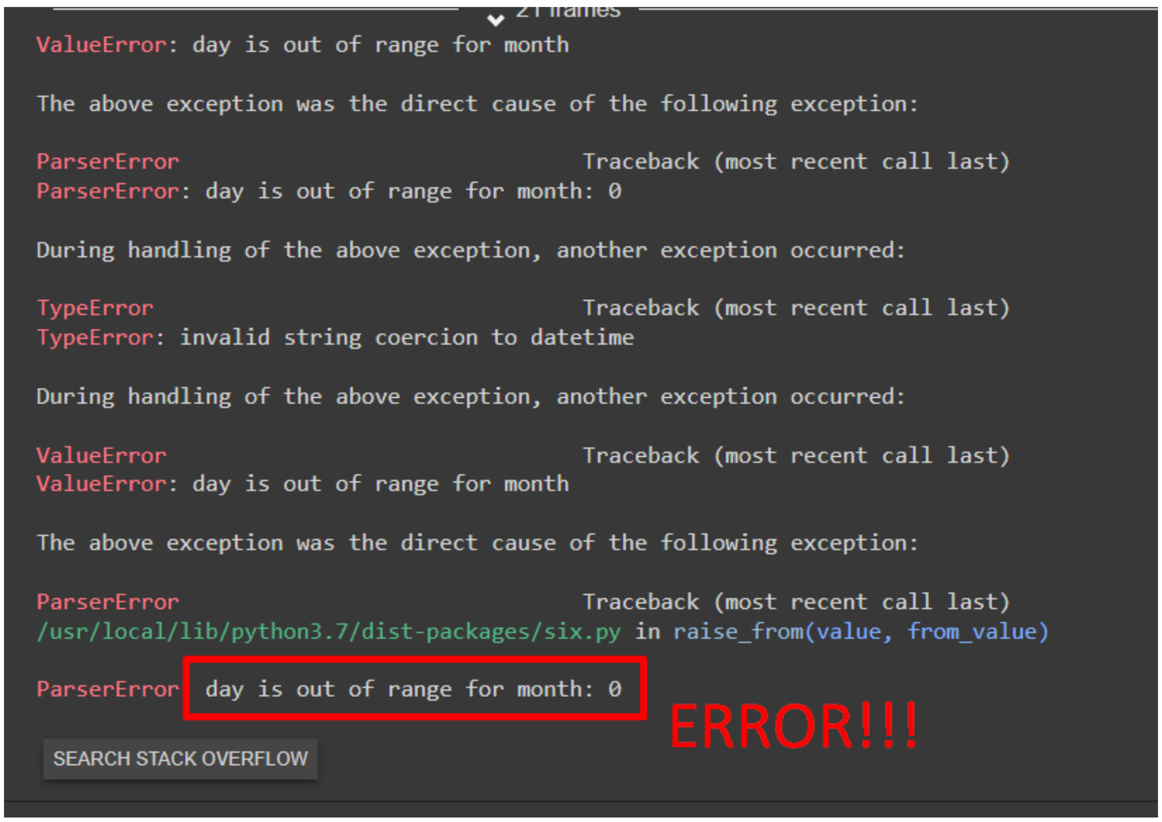
```
194    Sep-1999
19     Jan-2001
180    Sep-2006
85     Oct-2002
94     Dec-1996
Name: earliest_cr_line, dtype: object
```

C. ATTEMPTING TO CONVERT STRING TO "DATE TIME" FORMAT

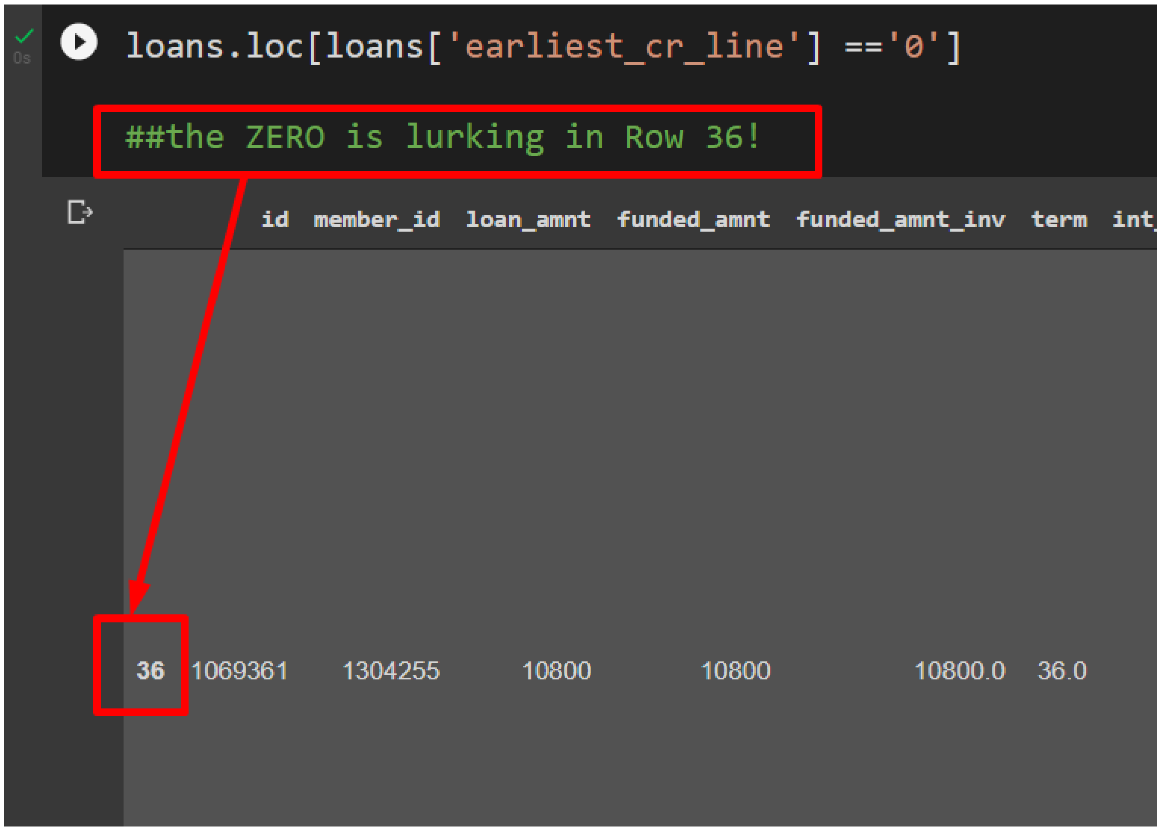


4c) Attempting to Convert String to "Date Time" Format

```
from datetime import datetime

loans['earliest_cr_line'] = \
    pd.to_datetime(
    loans['earliest_cr_line']
)

#we try converting directly to "datetime" format
#but an error pops up because there's a
#hidden 0 lurking somewhere....
```

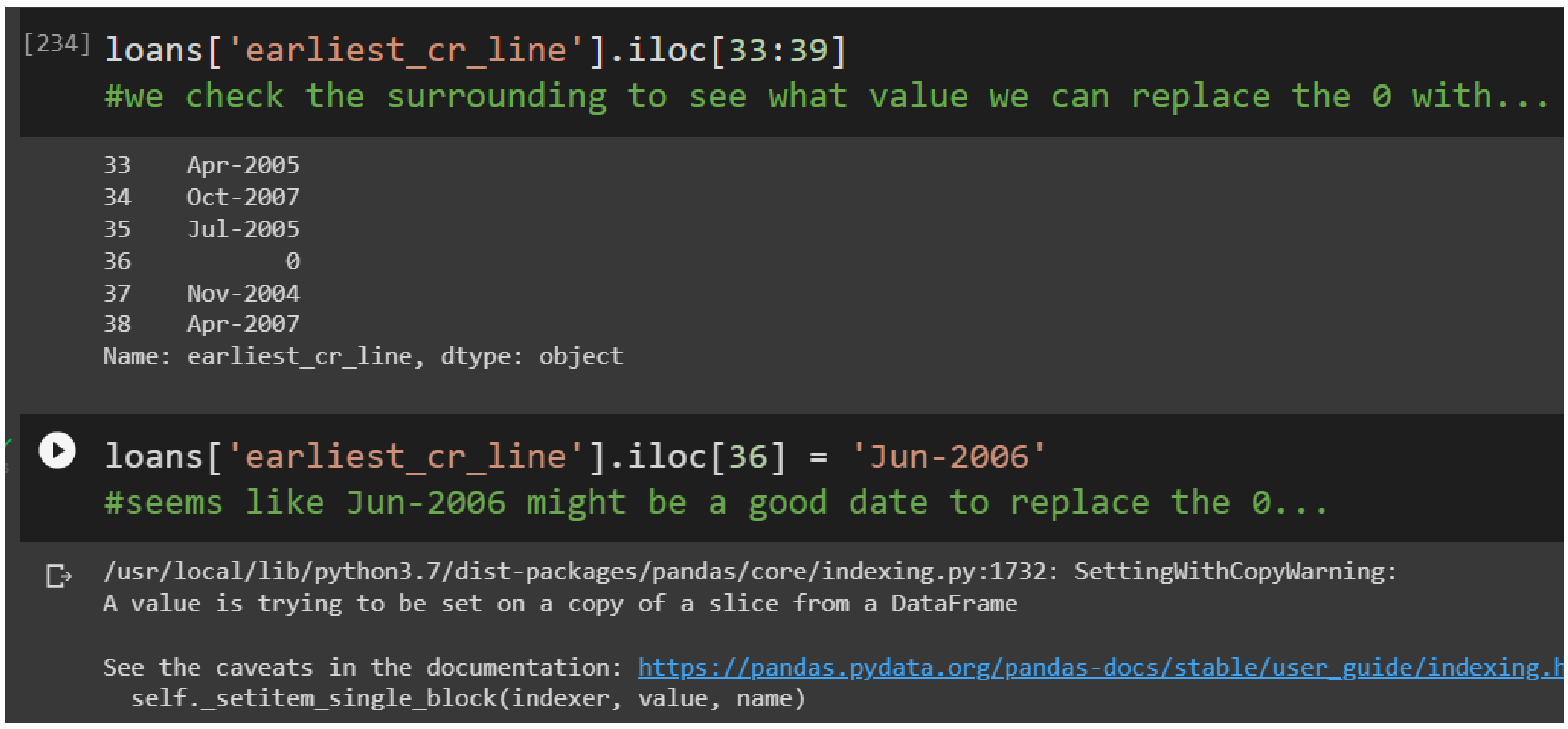


```
ValueError: day is out of range for month

The above exception was the direct cause of the following exception:

ParserError                               Traceback (most recent call last)
ParserError: day is out of range for month: 0

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
TypeError: invalid string coercion to datetime

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
ValueError: day is out of range for month

The above exception was the direct cause of the following exception:

ParserError                               Traceback (most recent call last)
/usr/local/lib/python3.7/dist-packages/six.py in raise_from(value, from_value)

ParserError: day is out of range for month: 0
```



```
loans.loc[loans['earliest_cr_line'] =='0']

##the ZERO is lurking in Row 36!
```

```
         id  member_id  loan_amnt  funded_amnt  funded_amnt_inv  term  int_
36  1069361    1304255      10800        10800          10800.0  36.0
```

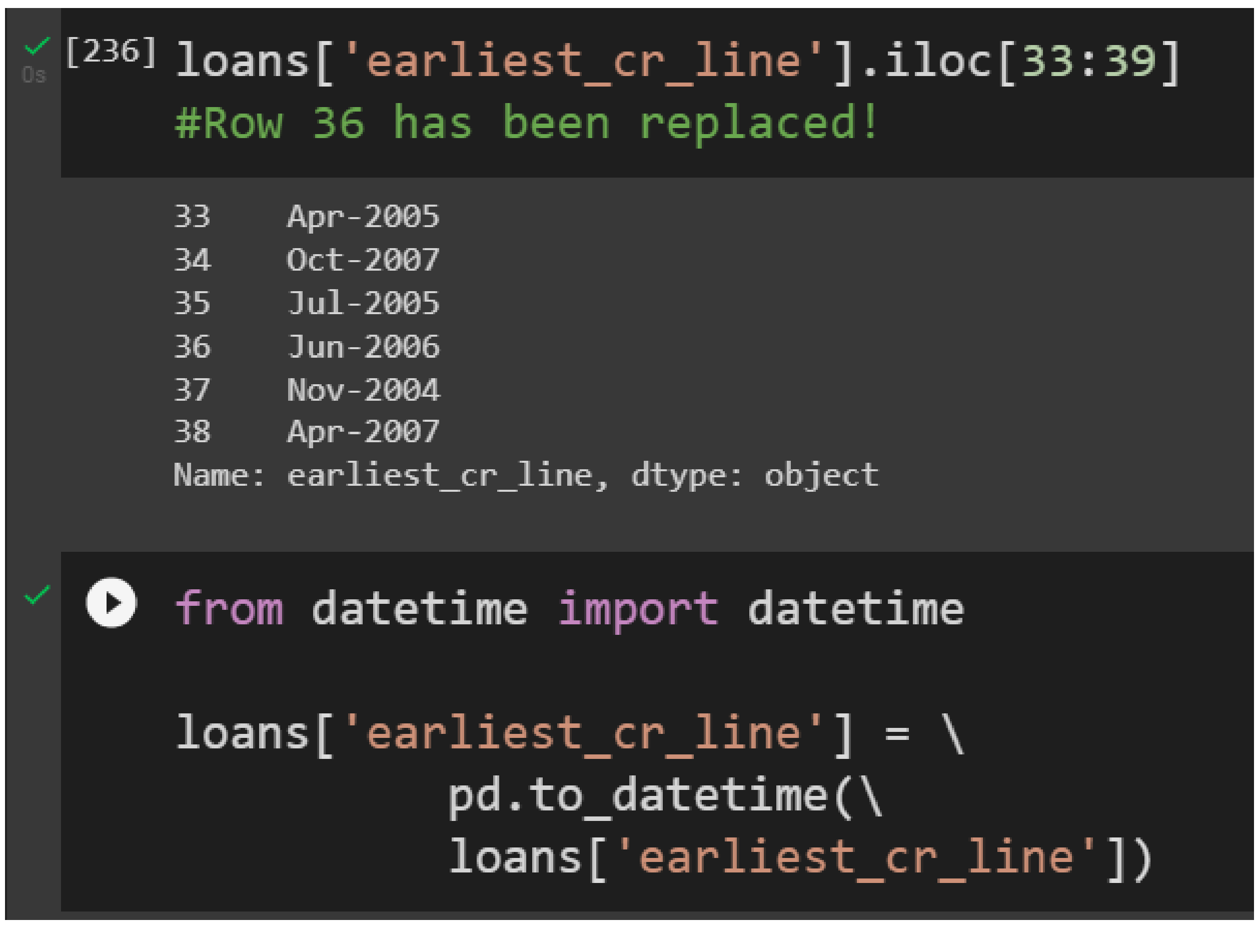


```
[234] loans['earliest_cr_line'].iloc[33:39]
      #we check the surrounding to see what value we can replace the 0 with...
```

```
33    Apr-2005
34    Oct-2007
35    Jul-2005
36           0
37    Nov-2004
38    Apr-2007
Name: earliest_cr_line, dtype: object
```

```
loans['earliest_cr_line'].iloc[36] = 'Jun-2006'
#seems like Jun-2006 might be a good date to replace the 0...
```

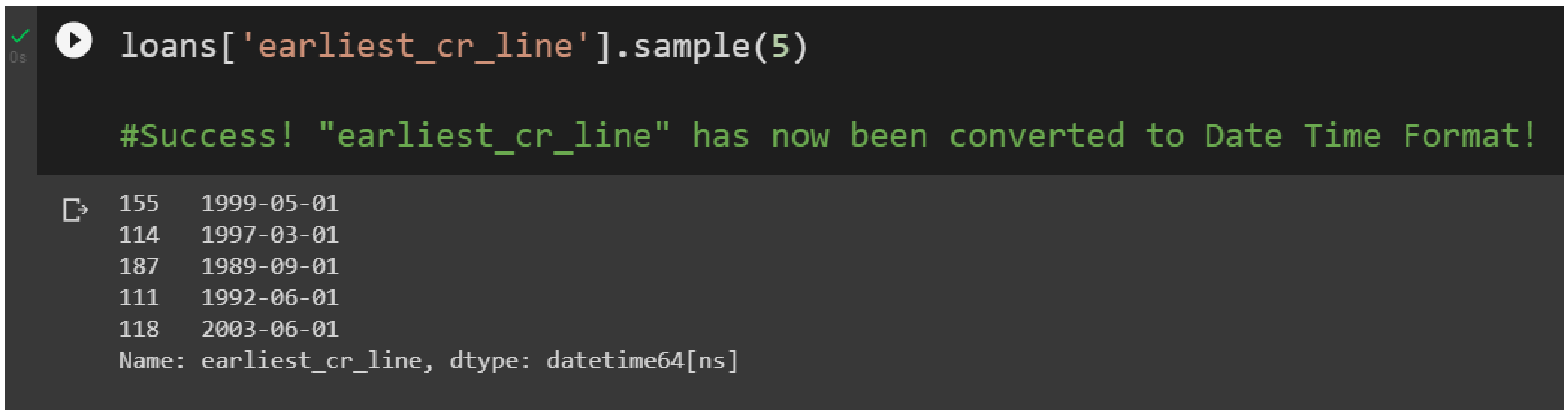
```
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1732: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
  self._setitem_single_block(indexer, value, name)
```



```
[236] loans['earliest_cr_line'].iloc[33:39]
      #Row 36 has been replaced!
```

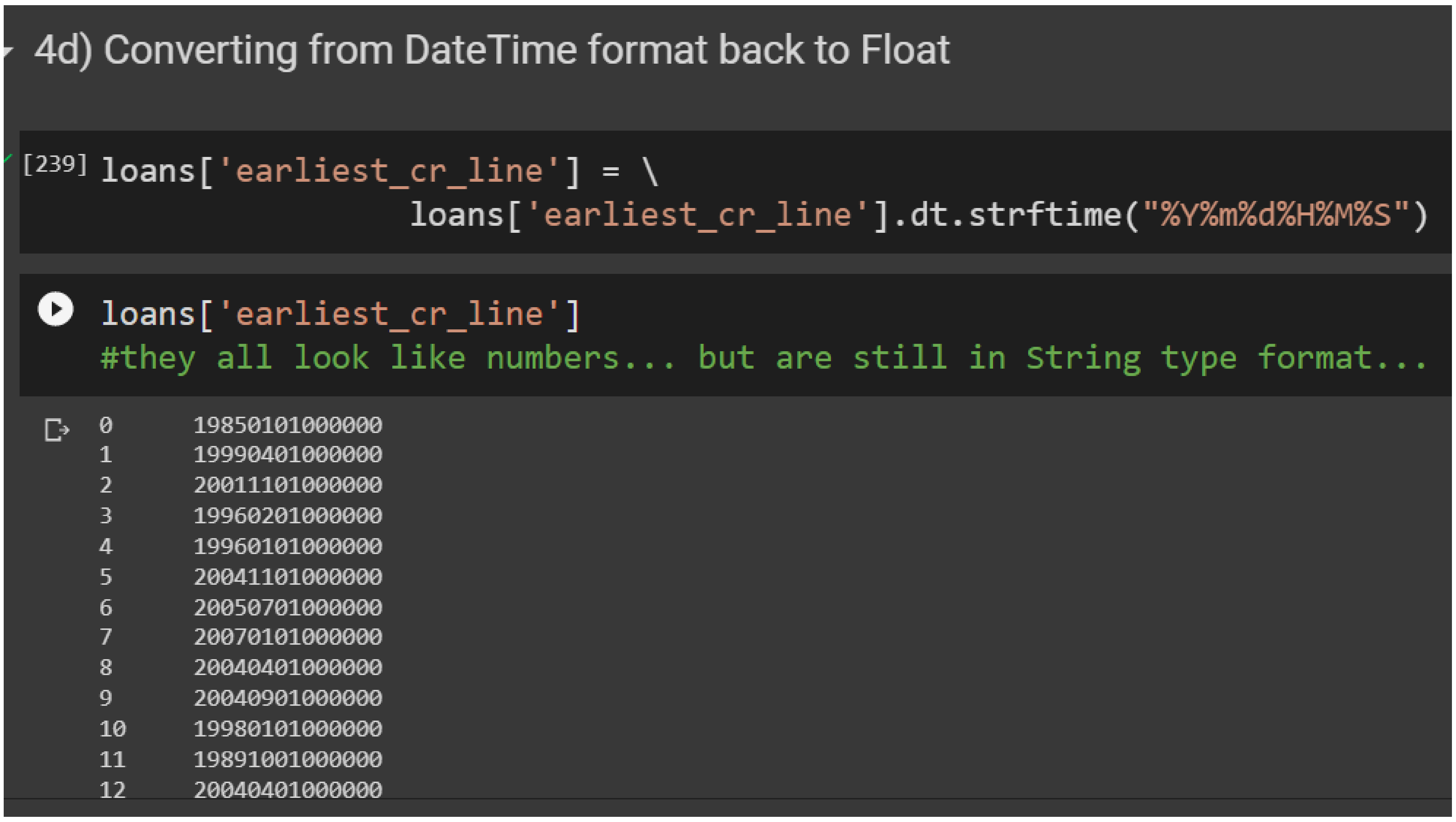
```
33    Apr-2005
34    Oct-2007
35    Jul-2005
36    Jun-2006
37    Nov-2004
38    Apr-2007
Name: earliest_cr_line, dtype: object
```

```
from datetime import datetime

loans['earliest_cr_line'] = \
            pd.to_datetime(\
            loans['earliest_cr_line'])
```



```
loans['earliest_cr_line'].sample(5)

#Success! "earliest_cr_line" has now been converted to Date Time Format!
```

```
155   1999-05-01
114   1997-03-01
187   1989-09-01
111   1992-06-01
118   2003-06-01
Name: earliest_cr_line, dtype: datetime64[ns]
```

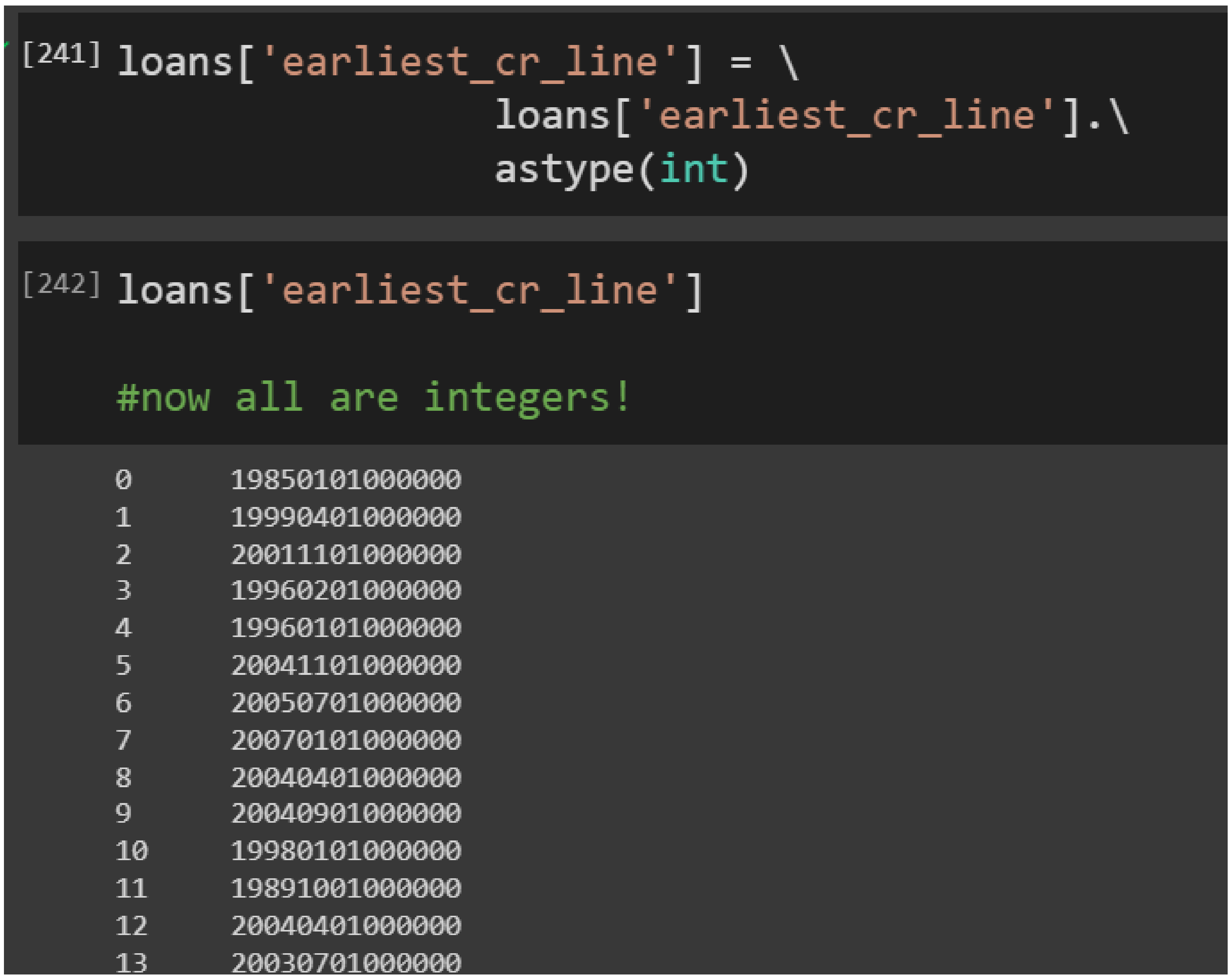


4d) Converting from DateTime format back to Float

```
[239] loans['earliest_cr_line'] = \
                          loans['earliest_cr_line'].dt.strftime("%Y%m%d%H%M%S")
```

```
loans['earliest_cr_line']
#they all look like numbers... but are still in String type format...
```

```
0       19850101000000
1       19990401000000
2       20011101000000
3       19960201000000
4       19960101000000
5       20041101000000
6       20050701000000
7       20070101000000
8       20040401000000
9       20040901000000
10      19980101000000
11      19891001000000
12      20040401000000
```



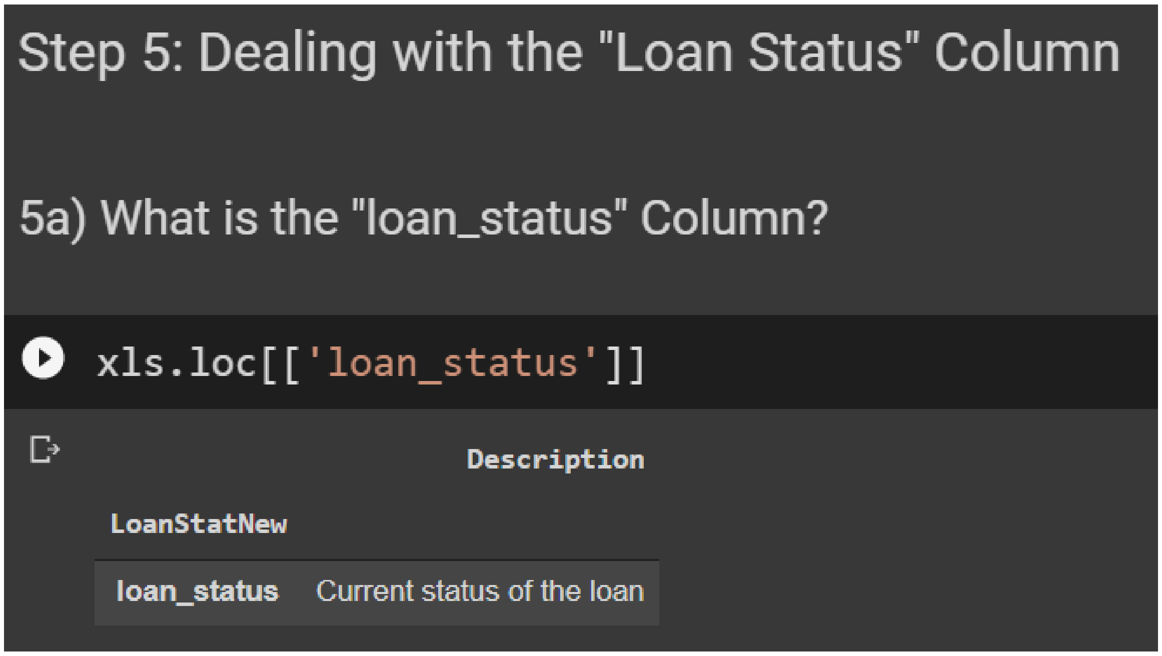
```
[241] loans['earliest_cr_line'] = \
                          loans['earliest_cr_line'].\
                          astype(int)
```

```
[242] loans['earliest_cr_line']

      #now all are integers!
```

```
0       19850101000000
1       19990401000000
2       20011101000000
3       19960201000000
4       19960101000000
5       20041101000000
6       20050701000000
7       20070101000000
8       20040401000000
9       20040901000000
10      19980101000000
11      19891001000000
12      20040401000000
13      20030701000000
```

I. STEP 5: DEALING WITH THE "LOAN STATUS" COLUMN

A. WHAT IS THE "LOAN STATUS" COLUMN?



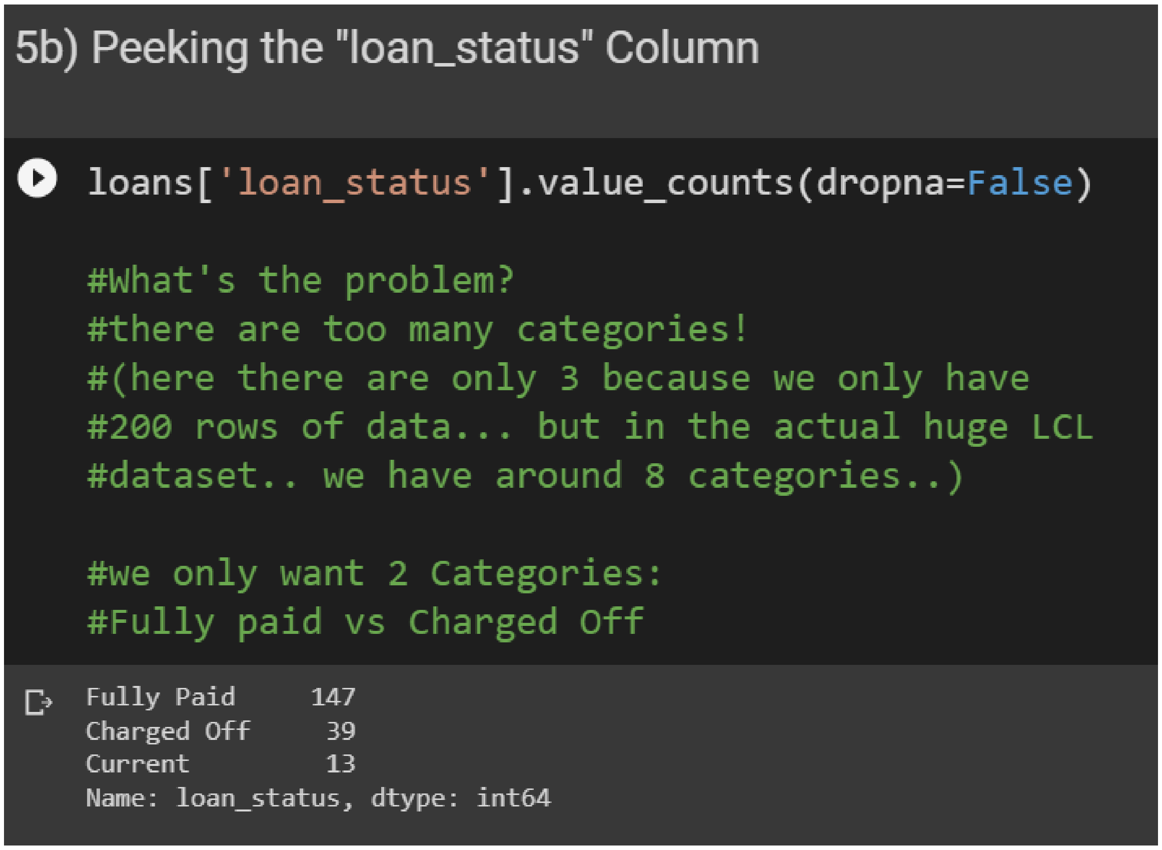
Step 5: Dealing with the "Loan Status" Column

5a) What is the "loan_status" Column?

```
xls.loc[['loan_status']]
```

```
             Description
LoanStatNew
loan_status  Current status of the loan
```

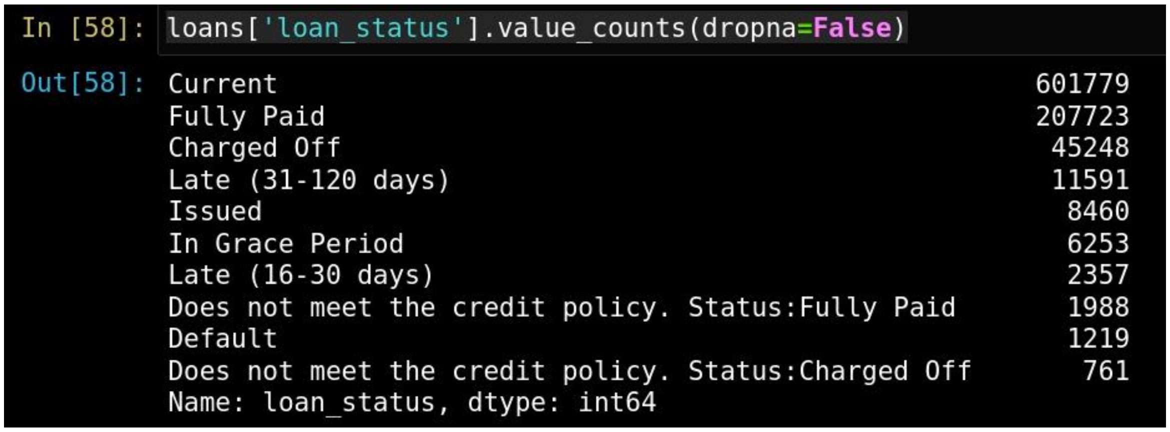
B. PEEKING AT THE "LOAN_STATUS" COLUMN



5b) Peeking the "loan_status" Column

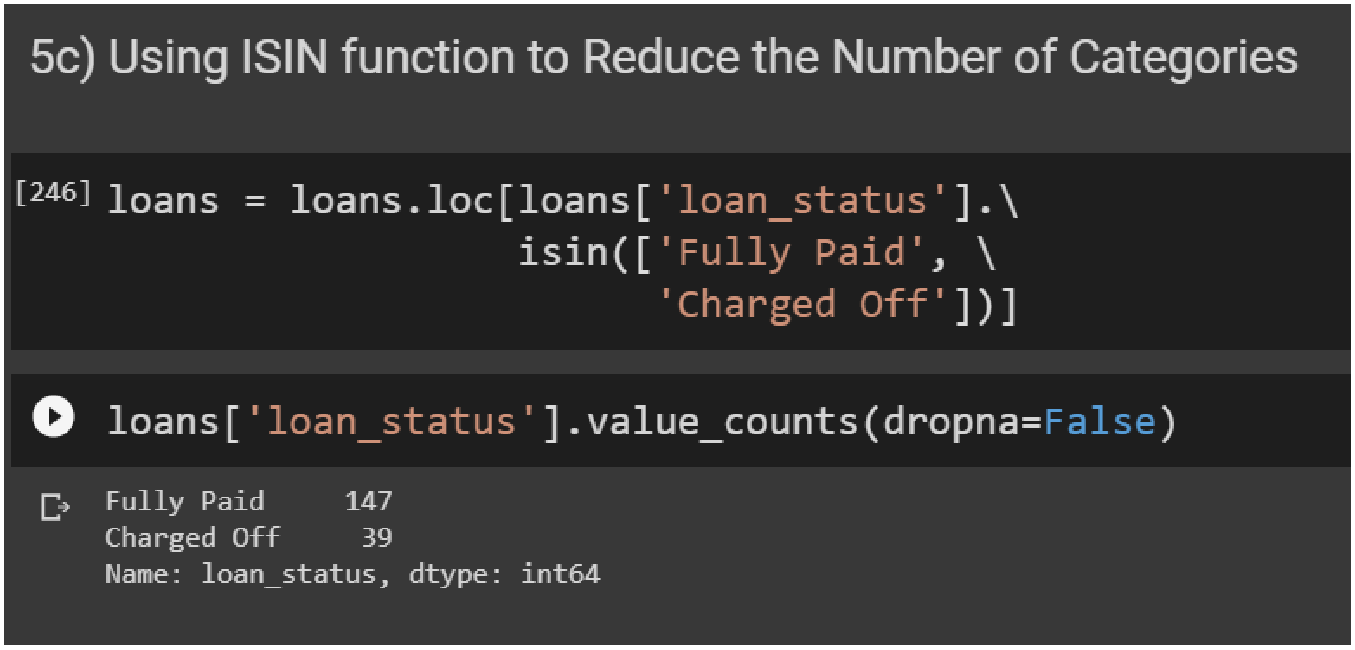
```
loans['loan_status'].value_counts(dropna=False)

#What's the problem?
#there are too many categories!
#(here there are only 3 because we only have
#200 rows of data... but in the actual huge LCL
#dataset.. we have around 8 categories..)

#we only want 2 Categories:
#Fully paid vs Charged Off
```

```
Fully Paid     147
Charged Off     39
Current         13
Name: loan_status, dtype: int64
```



```
In [58]: loans['loan_status'].value_counts(dropna=False)

Out[58]: Current                                                601779
         Fully Paid                                             207723
         Charged Off                                             45248
         Late (31-120 days)                                      11591
         Issued                                                   8460
         In Grace Period                                          6253
         Late (16-30 days)                                        2357
         Does not meet the credit policy. Status:Fully Paid       1988
         Default                                                  1219
         Does not meet the credit policy. Status:Charged Off       761
         Name: loan_status, dtype: int64
```

C. USING ISIN FUNCTION TO REDUCE THE NUMBER OF CATEGORIES



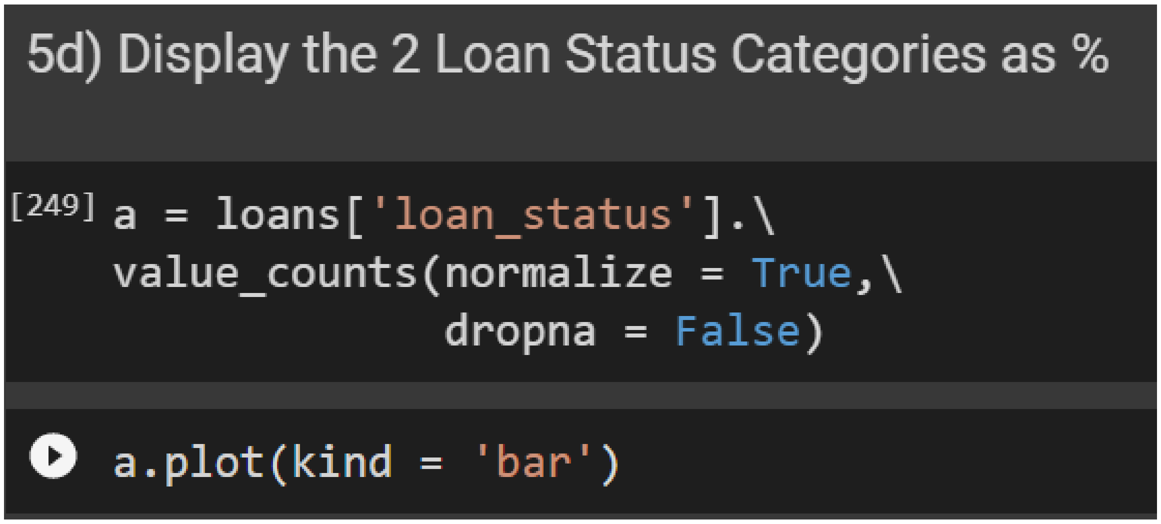
5c) Using ISIN function to Reduce the Number of Categories

```
[246] loans = loans.loc[loans['loan_status'].\
                          isin(['Fully Paid', \
                                'Charged Off'])]
```

```
loans['loan_status'].value_counts(dropna=False)
```

```
Fully Paid     147
Charged Off     39
Name: loan_status, dtype: int64
```

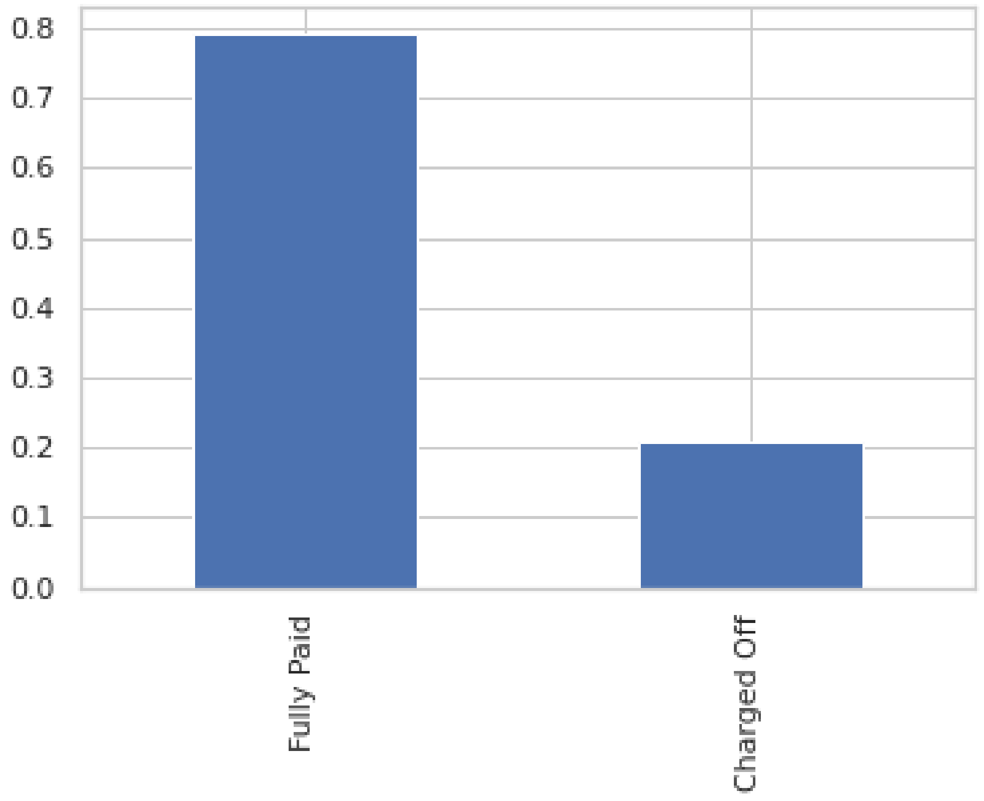
D. DISPLAY THE 2 LOAN STATUS CATEGORIES AS %



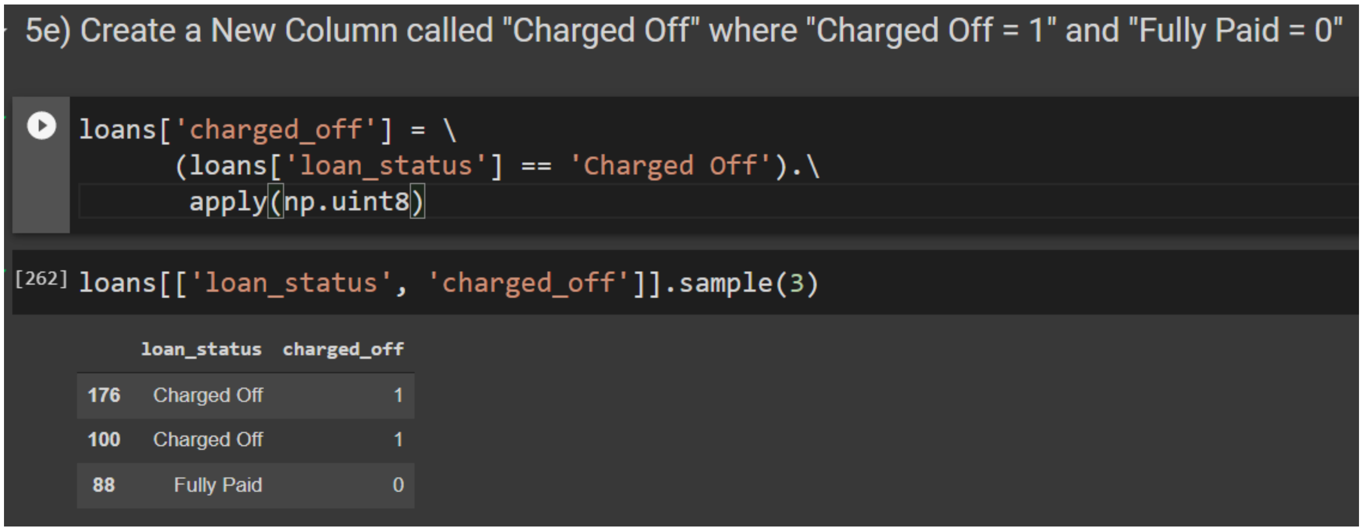
5d) Display the 2 Loan Status Categories as %

```
[249] a = loans['loan_status'].\
      value_counts(normalize = True,\
                   dropna = False)
```

```
a.plot(kind = 'bar')
```



E. CREATE A NEW COLUMN CALLED "CHARGED OFF" WHERE "CHARGED OFF = 1" AND "FULLY PAID = 0"



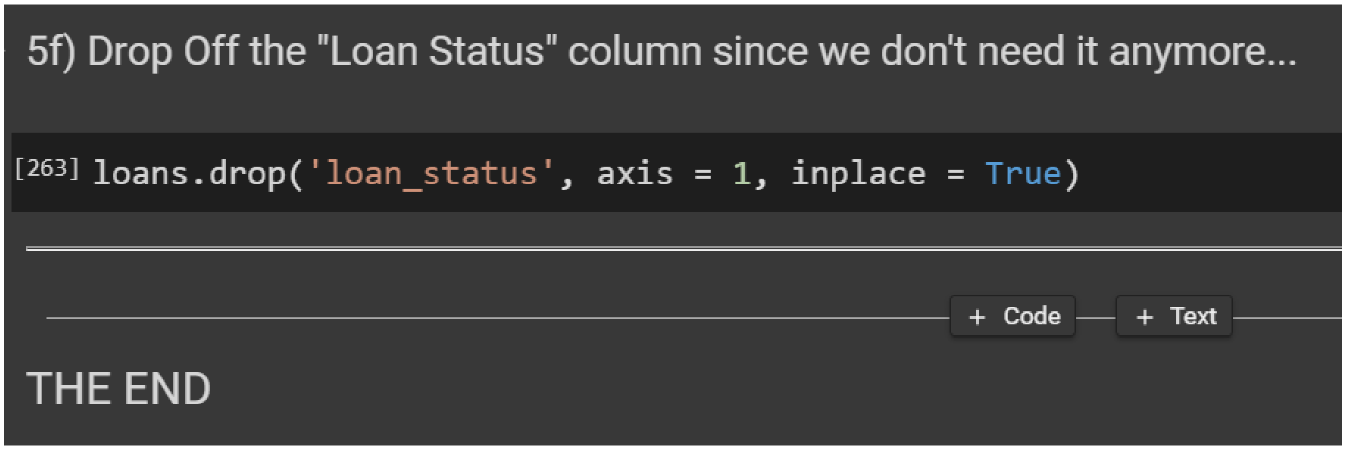
5e) Create a New Column called "Charged Off" where "Charged Off = 1" and "Fully Paid = 0"

```
loans['charged_off'] = \
      (loans['loan_status'] == 'Charged Off').\
       apply(np.uint8)
```

```
[262] loans[['loan_status', 'charged_off']].sample(3)
```

	loan_status	charged_off
176	Charged Off	1
100	Charged Off	1
88	Fully Paid	0

F. DROP OFF THE "LOAN STATUS" COLUMN SINCE WE DON'T NEED IT ANYMORE…



5f) Drop Off the "Loan Status" column since we don't need it anymore...

```
[263] loans.drop('loan_status', axis = 1, inplace = True)
```

THE END

ABOUT DR. ALVIN ANG

Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.