Springboard Data Science Career Track | Nikki Seegars

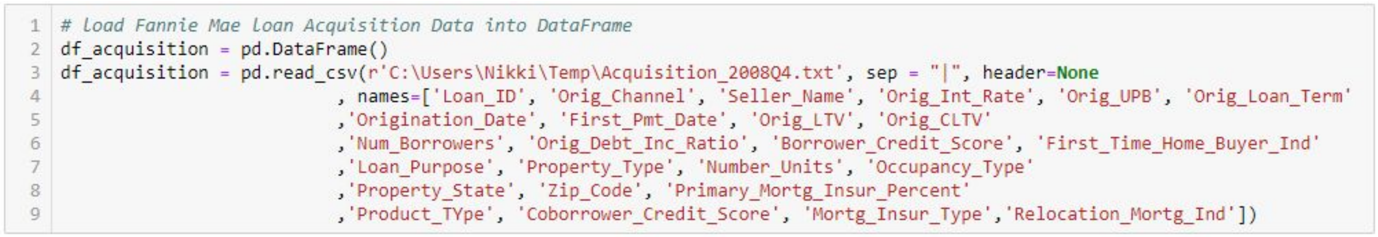
Capstone Project I: Data Wrangling

Data Collection

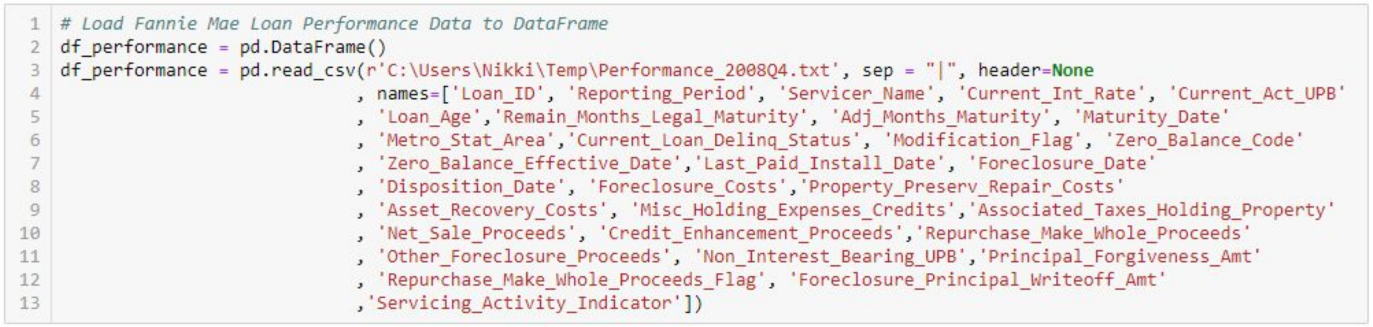
The dataset used for this project is from Fannie Mae single-family loan performance data which is housed on the website https://www.fanniemae.com/portal/funding-the-market/data/loan-performance-data.html. There are two text files for each quarter, one with acquisition data and another with performance data. The performance data files are very large, most being over 1GB in size. That being said, I chose to focus on one quarter, 4th quarter of 2008.

Loading into DataFrames

Upon reading the data into pandas dataframes, I used the resources on Fannie Mae's website to further understand the dataset. There are several documents outlining the structure of the dataset including column names and datatypes along with a FAQ document. Neither of the text files include a header row, so I needed to create the headers once I loaded the file into the dataframe.



```
# Load Fannie Mae Loan Acquisition Data into DataFrame
df_acquisition = pd.DataFrame()
df_acquisition = pd.read_csv(r'C:\Users\Nikki\Temp\Acquisition_2008Q4.txt', sep = "|", header=None
                             , names=['Loan_ID', 'Orig_Channel', 'Seller_Name', 'Orig_Int_Rate', 'Orig_UPB', 'Orig_Loan_Term'
                             ,'Origination_Date', 'First_Pmt_Date', 'Orig_LTV', 'Orig_CLTV'
                             ,'Num_Borrowers', 'Orig_Debt_Inc_Ratio', 'Borrower_Credit_Score', 'First_Time_Home_Buyer_Ind'
                             ,'Loan_Purpose', 'Property_Type', 'Number_Units', 'Occupancy_Type'
                             ,'Property_State', 'Zip_Code', 'Primary_Mortg_Insur_Percent'
                             ,'Product_TYpe', 'Coborrower_Credit_Score', 'Mortg_Insur_Type','Relocation_Mortg_Ind'])
```

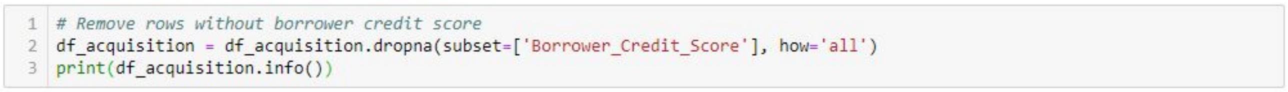


```
# Load Fannie Mae Loan Performance Data to DataFrame
df_performance = pd.DataFrame()
df_performance = pd.read_csv(r'C:\Users\Nikki\Temp\Performance_2008Q4.txt', sep = "|", header=None
                               , names=['Loan_ID', 'Reporting_Period', 'Servicer_Name', 'Current_Int_Rate', 'Current_Act_UPB'
                               , 'Loan_Age','Remain_Months_Legal_Maturity', 'Adj_Months_Maturity', 'Maturity_Date'
                               , 'Metro_Stat_Area','Current_Loan_Delinq_Status', 'Modification_Flag', 'Zero_Balance_Code'
                               , 'Zero_Balance_Effective_Date','Last_Paid_Install_Date', 'Foreclosure_Date'
                               , 'Disposition_Date', 'Foreclosure_Costs','Property_Preserv_Repair_Costs'
                               , 'Asset_Recovery_Costs', 'Misc_Holding_Expenses_Credits','Associated_Taxes_Holding_Property'
                               , 'Net_Sale_Proceeds', 'Credit_Enhancement_Proceeds','Repurchase_Make_Whole_Proceeds'
                               , 'Other_Foreclosure_Proceeds', 'Non_Interest_Bearing_UPB','Principal_Forgiveness_Amt'
                               , 'Repurchase_Make_Whole_Proceeds_Flag', 'Foreclosure_Principal_Writeoff_Amt'
                               ,'Servicing_Activity_Indicator'])
```

Data Inspection and Cleaning

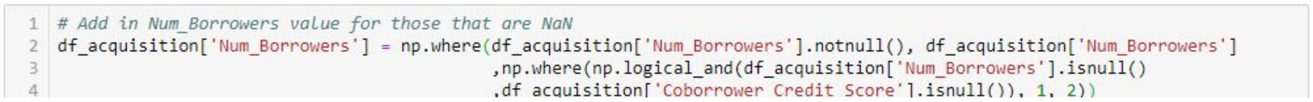
Once the data was loaded into DataFrames, I examined each of the columns by using looking at sample data using the pandas header() method and summary data by using pandas info() and memory_usage() methods. Each of the columns' datatypes are appropriate for the type of data used. The text data has the object data type and numeric data has either float64 or int64 datatype. I inspected each of the columns for any missing data and both of the text files were set up with all missing data as NaN.

Over 99% of the rows in the performance dataset had borrower credit score, however I dropped the small amount of rows that had NaN values.



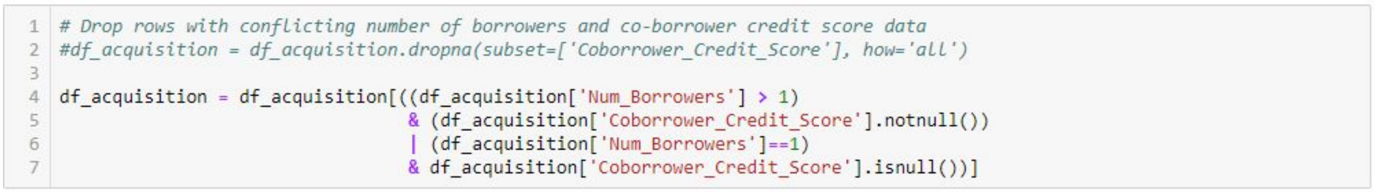
```
# Remove rows without borrower credit score
df_acquisition = df_acquisition.dropna(subset=['Borrower_Credit_Score'], how='all')
print(df_acquisition.info())
```

A small amount of loans, 25, are missing number of borrower values. I filled the values with either 1 or 2 depending on whether or not there was a co-borrower credit score for that record.



```
# Add in Num_Borrowers value for those that are NaN
df_acquisition['Num_Borrowers'] = np.where(df_acquisition['Num_Borrowers'].notnull(), df_acquisition['Num_Borrowers']
                                          ,np.where(np.logical_and(df_acquisition['Num_Borrowers'].isnull()
                                          ,df_acquisition['Coborrower_Credit_Score'].isnull()), 1, 2))
```

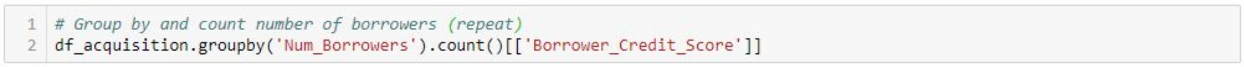
There were 9 records that had number of borrowers as 1 but had data in the co-borrower credit score column. There were 26,738 records that had number of borrowers > 1 with null values for the co-borrower credit score. I dropped the rows from the DataFrame for both of these scenarios.



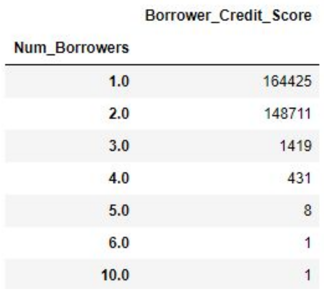
```
# Drop rows with conflicting number of borrowers and co-borrower credit score data
#df_acquisition = df_acquisition.dropna(subset=['Coborrower_Credit_Score'], how='all')

df_acquisition = df_acquisition[((df_acquisition['Num_Borrowers'] > 1)
                                & (df_acquisition['Coborrower_Credit_Score'].notnull())
                                | (df_acquisition['Num_Borrowers']==1)
                                & df_acquisition['Coborrower_Credit_Score'].isnull())]
```

The number of borrowers ranged from one to ten borrowers, with most having one or two borrowers. The dataset has two columns for credit score, one for primary borrower and another for co-borrower(s). I kept all the data and did not remove any outliers for the various number of borrowers since there is only one aggregate number for the co-borrowers' credit score.

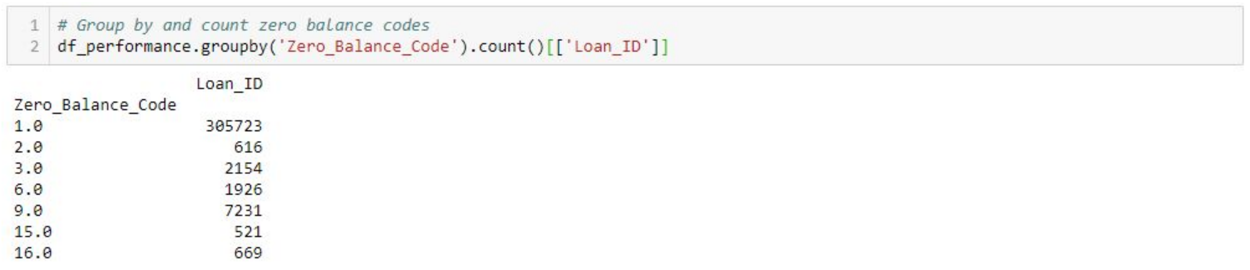


```
# Group by and count number of borrowers (repeat)
df_acquisition.groupby('Num_Borrowers').count()[['Borrower_Credit_Score']]
```



Num_Borrowers	Borrower_Credit_Score
1.0	164425
2.0	148711
3.0	1419
4.0	431
5.0	8
6.0	1
10.0	1

The performance data has a column for a zero balance code to identify closure reason for the account. The following are the reasons: 1 prepaid or matured, 2 third party sale, 3 short sale, 6 repurchased, 9 Deed-in-Lieu, REO, 15 new sale, 16 reperforming loan sale.



```python
# Group by and count zero balance codes
df_performance.groupby('Zero_Balance_Code').count()[['Loan_ID']]
```

```
                  Loan_ID
Zero_Balance_Code
1.0                305723
2.0                   616
3.0                  2154
6.0                  1926
9.0                  7231
15.0                  521
16.0                  669
```

Combining Data

The acquisition data was reduced from 342,128 to 314,996 records. The last step taken was to combine the acquisition and performance data on “Loan_ID”. The total number of records in the merged DataFrame is 12,965,819.