Data Wrangling Steps to Clean Dataset

Cleaning Steps: The data was gathered from a Kaggle competition so it was fairly clean. I was able to cleanly read in the .csv files and place then in a Pandas dataframe easily including row identifiers (each home is given an ID) and column names. Ran head(), tail(), shape(), info(), describe() and dtypes() methods on the dataframe to make sure that the data was fairly clean.

Did find columns with different but similar names. These were columns tamhog (size of the household) and column hhsize (household size). Column tamhog was dropped. Also, column agesq (age squared) and SQBage (age squared), agesq was dropped.

```
#compare colums hhsize = houlsehold size and tamhog = size of household
print(trainDF['hhsize'].equals(trainDF['tamhog']))
#compare columns agesq = 'Age squared' and SQBage = 'age squared'
print(trainDF['agesq'].equals(trainDF['SQBage']))
```

Found one column elimbasu5 (if rubbish disposal mainly by throwing in river, creek or sea) all these row values were 0 dropped by calculating the standard deviation.

```
#find only numeric columns and calculate standard deviation for each
numeric_cols = trainDF.select_dtypes(include=[np.number]).columns
colsStdDev = trainDF[numeric_cols].std()

#find any columns with a standard deviation of 0 and drop them from the #
#numeric_cols list
for index, value in colsStdDev.iteritems():
        if value == 0.0:
                templist.append(index)

#drop columns from trainDF that have a standard deviation of 0 (all row values for
column are same value)
trainDF = trainDF.drop(templist, axis=1)
```

Missing Values: Column v18q1 (number of tablets household owns) contained 7,928 rows with missing values out of 9,557 rows. This row was dropped as the values that remain were not significant enough to show any trends in data. Column rez_esc (years behind in school) had 7,028 missing values so this column was also dropped and column v2a1 (monthly rent) had 6,860 rows with missing values. Although this column seemed significant there was not enough substantial data to show a meaningful trend.

```
#includes drop of duplicate columns from above
trainDF = trainDF.drop(['agesq', 'tamhog', 'v18q1', 'rez_esc', 'v2a1'], axis=1)
```

For columns meaneduc (average years of education for adults 18+) and SQBmeaned (squared value of the meaneduc column) there were 5 rows with missing values in each column. These rows were dropped. Dropping the meaneduc rows also dropped the rows with missing values in SQBmeaned since they are dependent on each other.

```
#drop rows with missing values in meaneduc (average years of education for adults
#18+), and SQBmeaned (square of meanduc row). Dropping one will drop both
trainDF = trainDF.dropna(subset = ['meaneduc'])
```

Outliers: Separated only numeric values in the trainDF. Then separated the columns that only contained two values, 1 for yes and 0 for no. Calculated zscore for all columns that remained. Looked over the data to see if there were any erroneous values but everything looked like legitimate data.

```
#find only numeric columns and calculate standard deviation for each
numeric_cols = trainDF.select_dtypes(include=[np.number]).columns
#find columns that have more than 2 values (most columns contain a 1 value for yes
and a 0 value for no)
for column in numeric_cols:
        if (trainDF[column].value_counts(dropna=False).count() > 2):
                templist1.append(column)
#apply zscore to remaining columns
zscoreNumericCols = trainDF[templist1].apply(zscore)
#run a describe for columns
print(zscoreNumericCols.describe())
```