In [1]:
```
%load_ext autoreload
%autoreload 2
```

In [2]:
```
import pandas as pd
import numpy as np
```

1. Import HMDA data

- Data can be download from the CFPB site: 2021 LAR dataset
- The date the file was downloaded was appended to the raw file name.
- 99 Columns
- 26,204,358 records
- Data Dictionary

In [3]:
```
hmda21_df = pd.read_csv('C:\Temp\hmda2021_raw.csv', dtype = str)

hmda21_df
```

Out[3]:

	activity_year	lei	derived_msa_md	state_code	county_code	census_t
0	2021	549300MGPZBLQDIL7538	31084	CA	06037	06037554
1	2021	549300MGPZBLQDIL7538	32820	TN	47157	47157021
2	2021	549300MGPZBLQDIL7538	29404	IL	17097	17097864
3	2021	549300MGPZBLQDIL7538	99999	ND	38089	38089963
4	2021	549300MGPZBLQDIL7538	41700	TX	48029	48029131
...	...	...	...	...	...	
26204353	2021	549300MGPZBLQDIL7538	33124	FL	12086	1208601C
26204354	2021	549300MGPZBLQDIL7538	30980	TX	48203	4820302C
26204355	2021	549300MGPZBLQDIL7538	29820	NV	32003	32003005
26204356	2021	549300MGPZBLQDIL7538	99999	GA	13137	13137000
26204357	2021	549300MGPZBLQDIL7538	26900	IN	18081	1808161C

26204358 rows × 99 columns

In [4]:
```
hmda21_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26204358 entries, 0 to 26204357
Data columns (total 99 columns):
 #   Column                                    Dtype 
---  ------                                    ----- 
 0   activity_year                             object
 1   lei                                       object
 2   derived_msa_md                            object
 3   state_code                                object
 4   county_code                               object
 5   census_tract                              object
 6   conforming_loan_limit                     object
 7   derived_loan_product_type                 object
 8   derived_dwelling_category                 object
 9   derived_ethnicity                         object
 10  derived_race                              object
 11  derived_sex                               object
 12  action_taken                              object
 13  purchaser_type                            object
 14  preapproval                               object
 15  loan_type                                 object
 16  loan_purpose                              object
 17  lien_status                               object
 18  reverse_mortgage                          object
 19  open_end_line_of_credit                   object
 20  business_or_commercial_purpose            object
 21  loan_amount                               object
 22  combined_loan_to_value_ratio              object
 23  interest_rate                             object
 24  rate_spread                               object
 25  hoepa_status                              object
 26  total_loan_costs                          object
 27  total_points_and_fees                     object
 28  origination_charges                       object
 29  discount_points                           object
 30  lender_credits                            object
 31  loan_term                                 object
 32  prepayment_penalty_term                   object
 33  intro_rate_period                         object
 34  negative_amortization                     object
 35  interest_only_payment                     object
 36  balloon_payment                           object
 37  other_nonamortizing_features              object
 38  property_value                            object
 39  construction_method                       object
 40  occupancy_type                            object
 41  manufactured_home_secured_property_type   object
 42  manufactured_home_land_property_interest  object
 43  total_units                               object
 44  multifamily_affordable_units              object
 45  income                                    object
 46  debt_to_income_ratio                      object
 47  applicant_credit_score_type               object
 48  co_applicant_credit_score_type            object
 49  applicant_ethnicity_1                     object
 50  applicant_ethnicity_2                     object
 51  applicant_ethnicity_3                     object
 52  applicant_ethnicity_4                     object
 53  applicant_ethnicity_5                     object
 54  co_applicant_ethnicity_1                  object
```

```
 55  co_applicant_ethnicity_2                 object
 56  co_applicant_ethnicity_3                 object
 57  co_applicant_ethnicity_4                 object
 58  co_applicant_ethnicity_5                 object
 59  applicant_ethnicity_observed             object
 60  co_applicant_ethnicity_observed          object
 61  applicant_race_1                         object
 62  applicant_race_2                         object
 63  applicant_race_3                         object
 64  applicant_race_4                         object
 65  applicant_race_5                         object
 66  co_applicant_race_1                      object
 67  co_applicant_race_2                      object
 68  co_applicant_race_3                      object
 69  co_applicant_race_4                      object
 70  co_applicant_race_5                      object
 71  applicant_race_observed                  object
 72  co_applicant_race_observed               object
 73  applicant_sex                            object
 74  co_applicant_sex                         object
 75  applicant_sex_observed                   object
 76  co_applicant_sex_observed                object
 77  applicant_age                            object
 78  co_applicant_age                         object
 79  applicant_age_above_62                   object
 80  co_applicant_age_above_62                object
 81  submission_of_application                object
 82  initially_payable_to_institution         object
 83  aus_1                                    object
 84  aus_2                                    object
 85  aus_3                                    object
 86  aus_4                                    object
 87  aus_5                                    object
 88  denial_reason_1                          object
 89  denial_reason_2                          object
 90  denial_reason_3                          object
 91  denial_reason_4                          object
 92  tract_population                         object
 93  tract_minority_population_percent        object
 94  ffiec_msa_md_median_family_income        object
 95  tract_to_msa_income_percentage           object
 96  tract_owner_occupied_units               object
 97  tract_one_to_four_family_homes           object
 98  tract_median_age_of_housing_units        object
dtypes: object(99)
memory usage: 19.3+ GB
```

2. Clean Data

Dropping columns I don't need (21 in total) to make the data easier to work with:

The following columns were added by the CFPB, not using them.

- derived_loan_product_type
- derived_dwelling_category
- derived_ethnicity

- derived_race
- derived_sex

Focusing on the applicant's first ethnicity

- applicant_ethnicity-2
- applicant_ethnicity-3
- applicant_ethnicity-4
- applicant_ethnicity-5

Focusing on the co-applicant's first ethnicity. Don't need these columns to find co-applicants.

- co-applicant_ethnicity-2
- co-applicant_ethnicity-3
- co-applicant_ethnicity-4
- co-applicant_ethnicity-5

Focusing on the applicant's first race

- applicant_race-2
- applicant_race-3
- applicant_race-4
- applicant_race-5

Focusing on the co-applicant's first race. Don't need these columns to find co-applicants.

- co-applicant_race-2
- co-applicant_race-3
- co-applicant_race-4
- co-applicant_race-5

Using 78 columns instead of 99

In [5]:
```
remove_cols = ['derived_loan_product_type', 'derived_dwelling_category', 'derived_ethr
               'derived_race', 'derived_sex',
               'applicant_ethnicity_2','applicant_ethnicity_3', 'applicant_ethnicity_4
               'co_applicant_ethnicity_2', 'co_applicant_ethnicity_3', 'co_applicant_e
               'co_applicant_ethnicity_5',
               'applicant_race_2','applicant_race_3', 'applicant_race_4', 'applicant_r
               'co_applicant_race_2', 'co_applicant_race_3', 'co_applicant_race_4',
               'co_applicant_race_5']

new_headers = []
for column in hmda21_df.columns:
    if column not in remove_cols:
        new_headers.append(column)

print(len(new_headers))
```

```
78
```

Create smaller subset of HMDA data

- Deleting the orginal HMDA df to clear memory

In [6]:
```
hmda21_df2 = hmda21_df[new_headers].copy()
del hmda21_df
```

In [7]:
```
hmda21_df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26204358 entries, 0 to 26204357
Data columns (total 78 columns):
 #   Column                                    Dtype
---  ------                                    -----
 0   activity_year                             object
 1   lei                                       object
 2   derived_msa_md                            object
 3   state_code                                object
 4   county_code                               object
 5   census_tract                              object
 6   conforming_loan_limit                     object
 7   action_taken                              object
 8   purchaser_type                            object
 9   preapproval                               object
 10  loan_type                                 object
 11  loan_purpose                              object
 12  lien_status                               object
 13  reverse_mortgage                          object
 14  open_end_line_of_credit                   object
 15  business_or_commercial_purpose            object
 16  loan_amount                               object
 17  combined_loan_to_value_ratio              object
 18  interest_rate                             object
 19  rate_spread                               object
 20  hoepa_status                              object
 21  total_loan_costs                          object
 22  total_points_and_fees                     object
 23  origination_charges                       object
 24  discount_points                           object
 25  lender_credits                            object
 26  loan_term                                 object
 27  prepayment_penalty_term                   object
 28  intro_rate_period                         object
 29  negative_amortization                     object
 30  interest_only_payment                     object
 31  balloon_payment                           object
 32  other_nonamortizing_features              object
 33  property_value                            object
 34  construction_method                       object
 35  occupancy_type                            object
 36  manufactured_home_secured_property_type   object
 37  manufactured_home_land_property_interest  object
 38  total_units                               object
 39  multifamily_affordable_units              object
 40  income                                    object
 41  debt_to_income_ratio                      object
 42  applicant_credit_score_type               object
 43  co_applicant_credit_score_type            object
 44  applicant_ethnicity_1                     object
 45  co_applicant_ethnicity_1                  object
 46  applicant_ethnicity_observed              object
 47  co_applicant_ethnicity_observed           object
 48  applicant_race_1                          object
 49  co_applicant_race_1                       object
 50  applicant_race_observed                   object
 51  co_applicant_race_observed                object
 52  applicant_sex                             object
 53  co_applicant_sex                          object
 54  applicant_sex_observed                    object
```

```
 55  co_applicant_sex_observed                 object
 56  applicant_age                             object
 57  co_applicant_age                          object
 58  applicant_age_above_62                    object
 59  co_applicant_age_above_62                 object
 60  submission_of_application                 object
 61  initially_payable_to_institution          object
 62  aus_1                                     object
 63  aus_2                                     object
 64  aus_3                                     object
 65  aus_4                                     object
 66  aus_5                                     object
 67  denial_reason_1                           object
 68  denial_reason_2                           object
 69  denial_reason_3                           object
 70  denial_reason_4                           object
 71  tract_population                          object
 72  tract_minority_population_percent         object
 73  ffiec_msa_md_median_family_income         object
 74  tract_to_msa_income_percentage            object
 75  tract_owner_occupied_units                object
 76  tract_one_to_four_family_homes            object
 77  tract_median_age_of_housing_units         object
dtypes: object(78)
memory usage: 15.2+ GB
```

2. Clean Location

In [8]:
```
### Group all unique combinations of county codes and census tract
location_df = pd.DataFrame(hmda21_df2.groupby(by = ['county_code', 'census_tract'], dr
              reset_index().rename(columns = {0: 'count'})

### Replacing the Nulls with other text so that the function works, Keeping the Nulls
location_df = location_df.replace(to_replace = 'Na', value = 'ii-ii')
location_df = location_df.fillna('00-00')
```

In [9]:
```
### Number of unique combinations of county and census
print(len(location_df))

### Records where county code or census tract are Na
print(((location_df['county_code'] == 'ii-ii') | (location_df['census_tract'] == 'ii-i
### Records where county or census tract are NULL
print(((location_df['county_code'] == '00-00') | (location_df['census_tract'] == '00-0

location_df.sample(3, random_state = 303)
```

```
78069
0
5017
```

Out[9]:

	county_code	census_tract	count
33903	26065	26065002200	168
30838	24013	24013506102	455
68022	48469	48469000302	48

In [10]:
```
def clean_location(row):
```

```
    '''
    Standardizes census tract and county codes into one column.
    Choosing tract or county for records where they are missing one piece of info
    '''



    census_tract = row['census_tract']
    county = row['county_code']
    null = ['00-00', 'ii-ii']

    ### If there's tract info but no county, return tract
    if census_tract not in null and county in null:
        return census_tract

    ### If there's no tract info but there's county info, return county
    elif census_tract in null and county not in null:
        return county

    ### When tract and county are not null, return tract
    elif census_tract not in null and county not in null:
        return census_tract

    ### When tract and county are both null, return '-----'
    elif census_tract in null and county in null:
        return '-----'
```

In [11]:
```
### Running clean_location function to ensure every record has county code
location_df['location_code'] = location_df.apply(clean_location, axis = 1)

### Split location column for state and county fips codes
location_df['state_fips'] = location_df['location_code'].str[0:2]
location_df['county_fips'] = location_df['location_code'].str[2:5]

### Number of records with no county code and census tract information
nulls_df = location_df[(location_df['state_fips'] == '--') & (location_df['county_fips
print('Number of records with location nulls: ' + str(nulls_df['count'].sum()))

### Remove columns that are no longer needed
location_df2 = location_df.drop(columns = ['count', 'location_code'], axis = 1)
```

```
Number of records with location nulls: 285293
```

In [12]:
```
### Replace two dashes and three dashes data points with NaN
location_df2 = location_df2.replace(to_replace = '--', value = np.nan)
location_df2 = location_df2.replace(to_replace = '---', value = np.nan)

### Replace '00-00' and 'ii-ii' with the orginal data points to join back to the orgir
location_df2 = location_df2.replace(to_replace = '00-00', value = np.nan)
location_df2 = location_df2.replace(to_replace = 'ii-ii', value = 'Na')

location_df2.head(1)
```

Out[12]:

	county_code	census_tract	state_fips	county_fips
0	01001	01001020100	01	001

In [13]:
```
hmda21_df2 = pd.merge(hmda21_df2, location_df2, how = 'left', on = ['county_code', 'ce
```

```
nulls_records = (hmda21_df2['county_fips'].isnull() & hmda21_df2['state_fips'].isnull(

### This number matches the one from above:
print('Null Records that don\'t have fips data: ' + str((nulls_records)))
```

```
Null Records that don't have fips data: 285293
```

3. Clean Race and Ethnicity

- 1: Native American
- 2: Asian
- 3: Black
- 4: Pacific Islander
- 5: White
- 6: Latino
- 7: Race NA

In [14]:
```
### Group race and ethnicity for all unique combinations
main_race_eth = pd.DataFrame(hmda21_df2.groupby(by = ['applicant_race_1', 'applicant_e
                             dropna = False).size()).reset_index().rename(columns = {0

### Replace NAs with 000 for cleaning purposes
main_race_eth = main_race_eth.fillna('000')
print(len(main_race_eth))
main_race_eth.head(2)
```

```
168
```

Out[14]:

	applicant_race_1	applicant_ethnicity_1	count
0	1	1	49136
1	1	11	3468

In [16]:
```
def clean_race_ethnicity(row):

    '''
    Standardizing and merging race and ethnicity columns into one consistent column
    '''

    try:
        race = row['applicant_race_1']
        ethnicity = row['applicant_ethnicity_1']

    ### this is intended for co-applicants
    except KeyError:
        race = row['co_applicant_race_1']
        ethnicity = row['co_applicant_ethnicity_1']



    latinx = ['1', '11', '12', '13', '14']
    asian = ['2', '21', '22', '23', '24', '25', '26', '27']
    pac_islander = ['4', '41', '42', '43', '44']
    black = ['3']
```

```
    white = ['5']
    native = ['1']
    ethnicity_na = ['2', '3', '4', '000']
    race_na = ['6', '7', '-1', '000']

    ### Used for co-applicants
    ethnicity_nocoapp = ['5']
    race_nocoapp = ['8']


    ### Latinx
    if ethnicity in latinx:
        return '6'
    ### Black
    elif ethnicity not in latinx and race in black:
        return '3'
    ### Asian
    elif ethnicity not in latinx and race in asian:
        return '2'
    ### Pacific Islander
    elif ethnicity not in latinx and race in pac_islander:
        return '4'
    ### Native
    elif ethnicity not in latinx and race in native:
        return '1'
    ### White
    elif ethnicity not in latinx and race in white:
        return '5'
    # Race NA
    elif ethnicity in ethnicity_na and race in race_na:
        return '7'

    ### No Co-Applicants: Where both are no co-applicants OR where just one is no co-d
    elif (ethnicity in ethnicity_nocoapp and race in race_nocoapp) or (race in race_nc
        return '8'
```

In [17]:

```
### Apply clean_race_ethnicity function for the r/e dataframe
main_race_eth['app_race_ethnicity'] = main_race_eth.apply(clean_race_ethnicity, axis =

### Replace 000 with NaN to join back with HMDA data
main_race_eth = main_race_eth.replace(to_replace = '000', value = np.nan)
### Drop Count Column
main_race_eth = main_race_eth.drop(columns = ['count'], axis = 1)

hmda21_df2 = pd.merge(hmda21_df2, main_race_eth, how = 'left', on = ['applicant_race_1

hmda21_df2['app_race_ethnicity'].value_counts(dropna = False)
```

Out[17]:

```
5    13900141
7     6193399
6     2582745
3     1703222
2     1631345
1      141397
4       52109
Name: app_race_ethnicity, dtype: int64
```

4. Clean Co Race and Ethnicity

- 1: Native American
- 2: Asian
- 3: Black
- 4: Pacific Islander
- 5: White
- 6: Latino
- 7: Race NA
- 8: No Coapp

In [18]:
```
coapp_race_ethnicity = pd.DataFrame(hmda21_df2.groupby(by = ['co_applicant_race_1', 'c
                                    dropna = False).size()).reset_index().rename(colum

coapp_race_ethnicity = coapp_race_ethnicity.fillna('000')
coapp_race_ethnicity.head(1)
```

Out[18]:

	co_applicant_race_1	co_applicant_ethnicity_1	count
0	1	1	15484

In [19]:
```
### Using clean_race_ethnicity function for the coapp r/e dataframe, it has a no co-ap
coapp_race_ethnicity['coapp_race_ethnicity'] = coapp_race_ethnicity.apply(clean_race_e

coapp_race_ethnicity = coapp_race_ethnicity.drop(columns = ['count'], axis = 1)
coapp_race_ethnicity = coapp_race_ethnicity.replace(to_replace = '000', value = np.nar

hmda21_df2 = pd.merge(hmda21_df2, coapp_race_ethnicity, how = 'left',
                      on = ['co_applicant_race_1', 'co_applicant_ethnicity_1'])

hmda21_df2['coapp_race_ethnicity'].value_counts(dropna = False)
```

Out[19]:
```
8    14059024
5     6364494
7     3593269
6     1007749
2      680798
3      429228
1       45658
4       24138
Name: coapp_race_ethnicity, dtype: int64
```

6. Same or Different Race for Co-Applicant

- 1: Same
- 2: Difference
- 3: Not Applicable

In [20]:
```
### group all instances of main applicants and co-applicants races and ethnicities
coapp_same_race = pd.DataFrame(hmda21_df2.groupby(by = ['app_race_ethnicity', 'coapp_r
                               dropna = False).size()).reset_index().rename(columns =

coapp_same_race.sample(2, random_state = 303)
```

Out[20]:

	app_race_ethnicity	coapp_race_ethnicity	count
9	2	2	528406
51	7	4	736

In [22]:

```
def find_same_race(row):
    '''
    Looking at an applicant's and co-applicant's race to determine to same or differer
    '''

    app_race_ethnicity = row['app_race_ethnicity']
    coapp_race_ethnicity = row['coapp_race_ethnicity']

    race_yes = ['1', '2', '3', '4', '5', '6']
    race_na = ['7']
    no_coapp = ['8']


    ### Same Race: Where App's race is not 7 and co-app's is not 8 and race is the sam
    if app_race_ethnicity not in race_na and coapp_race_ethnicity not in no_coapp and
        return '1'

    ### Different Race:
    elif app_race_ethnicity not in race_na and coapp_race_ethnicity not in race_na + r
        return '2'

    ### Race NA
    elif (app_race_ethnicity in race_na and coapp_race_ethnicity in race_na) or (app_r
        return '3'

    ### No Co-Applicant
    elif coapp_race_ethnicity in no_coapp:
        return '4'
```

In [23]:

```
### Find records where applicant and co-applicant are the same
coapp_same_race['coapp_same_race'] = coapp_same_race.apply(find_same_race, axis = 1)

coapp_same_race = coapp_same_race.drop(columns = ['count'], axis = 1)
hmda21_df2 = pd.merge(hmda21_df2, coapp_same_race, how = 'left',
                      on = ['app_race_ethnicity', 'coapp_race_ethnicity'])

hmda21_df2['coapp_same_race'].value_counts(dropna = False)
```

Out[23]:

```
4    14059024
1     7574637
3     3696685
2      874012
Name: coapp_same_race, dtype: int64
```

7. Clean Credit Models

- 1: Equifax
- 2: Experian
- 3: TransUnion

- 4: Vantage
- 5: More than one
- 6: Other Model
- 7: Credit Na

In [24]:
```
credit_models = pd.DataFrame(hmda21_df2.groupby(by = ['applicant_credit_score_type'],
                dropna = False).size()).reset_index().rename(columns = {0: 'count'})

credit_models.head(1)
```

Out[24]:

	applicant_credit_score_type	count
0	1	5415202

In [25]:
```
def clean_credit_model(row):

    '''
    Standardizing credit model column
    '''

    equifax = ['1']
    experian = ['2']
    transunion = ['3', '4']
    vantage = ['5', '6']
    more_than_one = ['7']
    other_model = ['8']
    credit_na = ['9', '1111']

    credit_model = row['applicant_credit_score_type']


    if credit_model in equifax:
        return '1'
    elif credit_model in experian:
        return '2'
    elif credit_model in transunion:
        return '3'
    elif credit_model in vantage:
        return '4'
    elif credit_model in more_than_one:
        return '5'
    elif credit_model in other_model:
        return '6'
    elif credit_model in credit_na:
        return '7'
```

In [26]:
```
### Using function to standardize credit model
credit_models['app_credit_model'] = credit_models.apply(clean_credit_model, axis = 1)

credit_models = credit_models.drop(columns = ['count'], axis = 1)

hmda21_df2 = pd.merge(hmda21_df2, credit_models, how = 'left', on = ['applicant_credit

hmda21_df2['app_credit_model'].value_counts(dropna = False)
```

Out[26]:
```
7    10014444
1     5415202
3     4674300
2     4275679
6     1141418
5      561517
4      121798
Name: app_credit_model, dtype: int64
```

8. Find Co-Applicants

- 9999 in age means no co-applicant
- 8888 in age means no applicable

In [27]:
```
coapp_cols = ['coapp_race_ethnicity', 'co_applicant_sex', 'co_applicant_age', 'co_appl

coapp_comb_df = pd.DataFrame(hmda21_df2.groupby(by = coapp_cols, dropna = False).size(
                columns = {0: 'count'})

coapp_comb_df.head(1)
```

Out[27]:

	coapp_race_ethnicity	co_applicant_sex	co_applicant_age	co_applicant_credit_score_type	count
0	1	1	25-34	1	311

In [29]:
```
def find_coapplicants(row):

    '''
    Looking for co applicants within five columns
    '''

    ### Co-Applicants
    coapp_race = ['1', '2', '3', '4', '5', '6']
    coapp_sex = ['1', '2']
    coapp_age = ['<25', '25-34', '35-44', '45-54', '55-64', '65-74', '>74']
    coapp_credit = ['1', '2', '3', '4', '5', '6', '7', '8']

    ### NA Co-Applicants
    na_coapp_race = ['7']
    na_coapp_sex = ['3', '4', '6']
    na_coapp_age = ['8888']
    na_coapp_credit = ['9', '1111']

    ### No Co-Applicants
    nocoapp_race = ['8']
    nocoapp_sex = ['5']
    nocoapp_age = ['9999']
    nocoapp_credit = ['10']


    ### Co-Applicants Rows
    co_race = row['coapp_race_ethnicity']
    co_sex = row['co_applicant_sex']
    co_age = row['co_applicant_age']
    co_credit = row['co_applicant_credit_score_type']
```

```
    # CO APPLICANTS: 1
    ### Records where Race is known and the other fields are not "no co-applicant"
    ### race = y AND sex != n AND age != n AND credit != n (6003752 records)
    if co_race in coapp_race and co_sex not in nocoapp_sex and co_age not in nocoapp_a
        return '1'

    ### Records where Sex is known and the other fields are not "no co-applicant"
    ### race != n AND sex = y AND age != n AND credit != n (438837 records)
    elif co_race not in nocoapp_race and co_sex in coapp_sex and co_age not in nocoapp
        return '1'

    ### Records where Age is known and the other fields are not "no co-applicant"
    ### race != n AND sex != n AND age = y AND credit != n (643528 records)
    elif co_race not in nocoapp_race and co_sex not in nocoapp_sex and co_age in coapp
        return '1'

    ### Records where Credit is known and the other fields are not "no co-applicant"
    ### race != n AND sex != n AND age != n AND credit = y (99 records)
    elif co_race not in nocoapp_race and co_sex not in nocoapp_sex and co_age not in n
        return '1'

    ### Where race and sex have info, but age or credit are "no co-applicant"
    # race = y AND sex = y AND (age = n or credit = N) (20324 records)
    elif (co_race in coapp_race and co_sex in coapp_sex) and (co_age in nocoapp_age or
        return '1'

    ### Where race or sex have info and the other is NA, but age or credit are "no co-
    # ((r = na and sex = y) or (r = y and sex = na)) and (age = no or credit = no) (49
    elif ((co_race in na_coapp_race and co_sex in coapp_sex) or (co_race in coapp_race
    (co_age in nocoapp_age or co_credit in nocoapp_credit):
        return '1'

    # NO CO APPLICANTS: 2

    ### Where race is no and the rest are not "yes"
    #race = no AND sex != y AND age != y AND credit != y (9136951 records)
    elif co_race in nocoapp_race and co_sex not in coapp_sex and co_age not in coapp_a
        return '2'

    ### Where sex is no and the rest are not "yes"
    #race != y AND sex = n AND age != y AND credit != y (1234 records)
    elif co_race not in coapp_race and co_sex in nocoapp_sex and co_age not in coapp_a
        return '2'

    ### Where age is no and the rest are not "yes"
    #race != y AND sex != y AND age = n AND credit != y (131133 records)
    elif co_race not in coapp_race and co_sex not in coapp_sex and co_age in nocoapp_a
        return '2'

    ### Where credit is no and the rest are not "yes"
    #race != y AND sex != y AND age != y AND credit = n (44 records)
    elif co_race not in coapp_race and co_sex not in coapp_sex and co_age not in coapp
        return '2'

    ### NA CO-APPLICANTS:

    ### Where all columns are not applicable
    #race = na AND sex = na AND age = na AND credit = na (1143149 records)
    elif co_race in na_coapp_race and co_sex in na_coapp_sex and co_age in na_coapp_ag
```

```
        return '3'

    ### Where race and sex, or at least one of them are no, and age or credit are yes
    # ((race = n and sex = n) or (race = n and sex = na) or (race = na and sex = n))ar
    elif ((co_race in nocoapp_race and co_sex in nocoapp_sex) or (co_race in nocoapp_r
    (co_race in na_coapp_race and co_sex in nocoapp_sex)) and (co_age in coapp_age or
        return '3'

    ### Where race and sex contradict each other
    # (race = y and sex = n) or (race = n and sex = y) (3450 records)
    elif (co_race in coapp_race and co_sex in nocoapp_sex) or (co_race in nocoapp_race
        return '3'

    ### Where race and sex are not applicable and age and credit contradict each other
    # (race = na and sex = na) and ((age = y and credit = n) or (age = n and credit =
    elif (co_race in na_coapp_race and co_sex in na_coapp_sex) and ((co_age in coapp_a
    (co_age in nocoapp_age and co_credit in coapp_credit)):
        return '3'
```

Co-Applicants

- 1: Co-Applicants
- 2: No co-applicants
- 3: Not Applicable

In [30]:
```
### Run function to find co-applicants
coapp_comb_df['co_applicant'] = coapp_comb_df.apply(find_coapplicants, axis = 1)

coapp_comb_df = coapp_comb_df.drop(columns = ['count'], axis = 1)

hmda21_df2 = pd.merge(hmda21_df2, coapp_comb_df, how = 'left', on = coapp_cols)

hmda21_df2['co_applicant'].value_counts(dropna = False)
```

Out[30]:
```
2    14059462
1    10339009
3     1805887
Name: co_applicant, dtype: int64
```

9. Standardize Outcomes

- 1: Loan originated
- 2: Application approved but not accepted
- 3: Application denied
- 4: Application withdrawn by applicant
- 5: File closed for incompleteness
- 6: Purchased loan
- 7: Preapproval request denied
- 8: Preapproval request approved but not accepted

In [31]:
```
action_taken = pd.DataFrame(hmda21_df2['action_taken'].value_counts(dropna = False)).r
               rename(columns = {'index': 'action_taken', 'action_taken': 'count'})
```

```
action_taken.head(2)
```

Out[31]:

	action_taken	count
0	1	15058625
1	4	3310502

In [32]:

```
def clean_outcomes(row):

    '''
    Standardize outcomes, grouping the ambiguous outcomes together
    '''

    outcome = row['action_taken']
    other_outcomes = ['2', '4', '5', '7', '8']

    ### Loans
    if outcome == '1':
        return '1'

    ### Denials
    elif outcome == '3':
        return '3'

    ### Othercomes
    elif outcome in other_outcomes:
        return '4'

    ### Purchase loans
    elif outcome == '6':
        return '6'
```

In [ ]:

Outcomes:

- 1: Loans
- 3: Denials
- 4: Other Outcomes
- 6: Purchase loans

In [33]:

```
### Clean Outcomes
action_taken['loan_outcome'] = action_taken.apply(clean_outcomes, axis = 1)

action_taken = action_taken.drop(columns = ['count'], axis = 1)

hmda21_df2 = pd.merge(hmda21_df2, action_taken, how = 'left', on = ['action_taken'])

hmda21_df2['loan_outcome'].value_counts(dropna = False)
```

Out[33]:

```
1    15058625
4     5543757
3     2921003
6     2680973
Name: loan_outcome, dtype: int64
```

10. Standardize Automated Underwriting System

In [34]:
```
aus = ['aus_1', 'aus_2', 'aus_3', 'aus_4', 'aus_5']

### Group all unique combinations of AUS together to find all the patterns
aus_df = pd.DataFrame(hmda21_df2.groupby(by = aus, dropna = False).size()).\
         reset_index().rename(columns = {0: 'count'})
aus_df = aus_df.drop(columns = ['count'], axis = 1)

aus_df.head(2)
```

Out[34]:

	aus_1	aus_2	aus_3	aus_4	aus_5
0	1	1	1	1	1
1	1	1	1	1	2

In [35]:
```
def find_aus_patterns(df):


    '''
    Looking for the patterns within the five aus columns
    Answering questions: How many times was the same aus used or were different ones u
    '''


    df_container = []

    for index_num in df.index:

        ### take a single row
        single_row_df = df.loc[[index_num]].copy()

        ### convert to series
        row = pd.Series(single_row_df.values[0])
        ### Count the values in that series

        valuescount_df = pd.DataFrame(row.value_counts(dropna = False))
        num_unqiue_values = valuescount_df.index.nunique(dropna = False)

        try:
            number_nulls = valuescount_df[(valuescount_df.index.isnull())].values[0][0
        except IndexError:
            number_nulls = 0

        single_row_df['number_of_values'] = num_unqiue_values
        single_row_df['number_of_nulls'] = number_nulls
        df_container.append(single_row_df)

    df = pd.concat(df_container)

    return df
```

In [36]:
```
def clean_aus(row):

    '''
    Created a standardized column for AUS
```

```
    '''


    aus1 = row['aus_1']
    unique_values = row['number_of_values']
    nulls = row['number_of_nulls']

    ### Only AUS was used
    if aus1 != '1111' and unique_values == 2 and nulls == 4:
        return '1'

    ### Same AUS used multiple times
    elif (unique_values == 2 and (nulls > 0 and nulls < 4)) or (unique_values == 1 and
        return '2'

    ### Different AUS used
    elif (unique_values >= 2 and nulls == 0) or (unique_values >= 3 and nulls <= 3):
        return '3'

    ## Exempt
    elif aus1 == '1111':
        return '4'
```

Write out new csv

In [38]:
```
hmda21_df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26204358 entries, 0 to 26204357
Data columns (total 86 columns):
 #   Column                                    Dtype 
---  ------                                    ----- 
 0   activity_year                             object
 1   lei                                       object
 2   derived_msa_md                            object
 3   state_code                                object
 4   county_code                               object
 5   census_tract                              object
 6   conforming_loan_limit                     object
 7   action_taken                              object
 8   purchaser_type                            object
 9   preapproval                               object
 10  loan_type                                 object
 11  loan_purpose                              object
 12  lien_status                               object
 13  reverse_mortgage                          object
 14  open_end_line_of_credit                   object
 15  business_or_commercial_purpose            object
 16  loan_amount                               object
 17  combined_loan_to_value_ratio              object
 18  interest_rate                             object
 19  rate_spread                               object
 20  hoepa_status                              object
 21  total_loan_costs                          object
 22  total_points_and_fees                     object
 23  origination_charges                       object
 24  discount_points                           object
 25  lender_credits                            object
 26  loan_term                                 object
 27  prepayment_penalty_term                   object
 28  intro_rate_period                         object
 29  negative_amortization                     object
 30  interest_only_payment                     object
 31  balloon_payment                           object
 32  other_nonamortizing_features              object
 33  property_value                            object
 34  construction_method                       object
 35  occupancy_type                            object
 36  manufactured_home_secured_property_type   object
 37  manufactured_home_land_property_interest  object
 38  total_units                               object
 39  multifamily_affordable_units              object
 40  income                                    object
 41  debt_to_income_ratio                      object
 42  applicant_credit_score_type               object
 43  co_applicant_credit_score_type            object
 44  applicant_ethnicity_1                     object
 45  co_applicant_ethnicity_1                  object
 46  applicant_ethnicity_observed              object
 47  co_applicant_ethnicity_observed           object
 48  applicant_race_1                          object
 49  co_applicant_race_1                       object
 50  applicant_race_observed                   object
 51  co_applicant_race_observed                object
 52  applicant_sex                             object
 53  co_applicant_sex                          object
 54  applicant_sex_observed                    object
```

```
 55  co_applicant_sex_observed                 object
 56  applicant_age                             object
 57  co_applicant_age                          object
 58  applicant_age_above_62                    object
 59  co_applicant_age_above_62                 object
 60  submission_of_application                 object
 61  initially_payable_to_institution          object
 62  aus_1                                     object
 63  aus_2                                     object
 64  aus_3                                     object
 65  aus_4                                     object
 66  aus_5                                     object
 67  denial_reason_1                           object
 68  denial_reason_2                           object
 69  denial_reason_3                           object
 70  denial_reason_4                           object
 71  tract_population                          object
 72  tract_minority_population_percent         object
 73  ffiec_msa_md_median_family_income         object
 74  tract_to_msa_income_percentage            object
 75  tract_owner_occupied_units                object
 76  tract_one_to_four_family_homes            object
 77  tract_median_age_of_housing_units         object
 78  state_fips                                object
 79  county_fips                               object
 80  app_race_ethnicity                        object
 81  coapp_race_ethnicity                      object
 82  coapp_same_race                           object
 83  app_credit_model                          object
 84  co_applicant                              object
 85  loan_outcome                              object
dtypes: object(86)
memory usage: 17.0+ GB
```

In [39]:
```
hmda21_df2
```

Out[39]:

	activity_year	lei	derived_msa_md	state_code	county_code	census_t
0	2021	549300MGPZBLQDIL7538	31084	CA	06037	06037554
1	2021	549300MGPZBLQDIL7538	32820	TN	47157	47157021
2	2021	549300MGPZBLQDIL7538	29404	IL	17097	17097864
3	2021	549300MGPZBLQDIL7538	99999	ND	38089	38089963
4	2021	549300MGPZBLQDIL7538	41700	TX	48029	48029131
...	...	...	...	...	...	
26204353	2021	549300MGPZBLQDIL7538	33124	FL	12086	1208601C
26204354	2021	549300MGPZBLQDIL7538	30980	TX	48203	4820302C
26204355	2021	549300MGPZBLQDIL7538	29820	NV	32003	32003005
26204356	2021	549300MGPZBLQDIL7538	99999	GA	13137	1313700C
26204357	2021	549300MGPZBLQDIL7538	26900	IN	18081	1808161C

26204358 rows × 86 columns

In [41]:
```
hmda21_df2.to_csv('C:\Temp\hmda21_cleaned.csv', index = False)
```

In [42]:
```
del(hmda21_df2)
```

In [ ]:
```

```