

Team 14: David Loving | Ilan Dor | Volodymyr Popil

Data Collection

```
In [1]: # import libraries
import pandas as pd
import numpy as np
import requests
from bs4 import BeautifulSoup
import time
```

Scrape FBI data

```
In [2]: # create array of years
years = np.arange(2006,2017,1)
```

```
In [3]: def fetch_fbi_year_data(year):
    '''
    Scrapes the FBI murder rate data by MSA from the web pages

    Inputs:
    --year, a year for which data should be scraped
    '''

    # dataframe to store the data
    df = pd.DataFrame()

    # get url of webpage with MSA data, links vary accross years
    if year < 2010:
        url = "https://www2.fbi.gov/ucr/cius" + str(year) + "/data/table_06.html"
    elif year in [2012,2013] :
        url = "https://ucr.fbi.gov/crime-in-the-u.s/" + str(year) + "/crime-in-the-u.s.-" \
            + str(year) + "/tables/6tabledatadecpdf/table-6"
```

```
elif year < 2016:
    url = "https://ucr.fbi.gov/crime-in-the-u.s/"\
        + str(year) + "/crime-in-the-u.s.-" + str(year) + "/tables/table-6"
else:
    url = "https://ucr.fbi.gov/crime-in-the-u.s/" + str(year) + "/crime-in-the-u.s.-"\
        + str(year) + "/topic-pages/tables/table-4"

# get HTML from the page and check the response code
response = requests.get(url)

# proceed if 200 response
if response.ok:
    # create instance of BeautifulSoup from the response text
    soup = BeautifulSoup(response.text, "html.parser")

    # fetch the first table matching class criteria
    table = soup.find("table", {"class": "data"})

    # get the table body
    tbody = table.find("tbody")

    # fetch all rows from the table
    rows = tbody.find_all("tr")

    # create list to store MSAs
    msa_murders = []

    # create dictionary to store MSAs and rates
    d = {}

    # iterate over rows
    for row in rows:

        # get header line (MSA)
        header_line = row.find("th", {"class": "subguide1"})\
            or row.find("th", {"class": "subguide2"})\
            or row.find("th", {"class": "even group0 alignleft valignmenttop"})\
```

```

        or row.find("th", {"class":"even group0 bold alignleft valignmenttop"})\
        or row.find("th", {"class":"even group0 bold valignmenttop"})

# store MSA if found
if header_line:
    msa = header_line.text
    msa = msa.replace("≡", "-")
    msa = msa.replace("\n", " ")
    msa = msa.strip()
    msa = msa[:msa.find(" M.S.A.")]

    # update dict
    d.update({"MSA":msa})
else:
    # var to store murder rate
    murder_per_100_k = 0

    # get the table entry
    line = row.find("th", {"class":"subguidela"})\
        or row.find("th", {"class":"subguidele"})\
        or row.find("th", {"class":"odd group1 alignleft valignmentbottom"})\
        or row.find("th", {"class":"odd group1 valignmentbottom"})

    line_label = ""

    if line:
        line_label = line.text

    # if match the criteria, store rate
    if line_label.strip() == "Rate per 100,000 inhabitants":
        # set custom index position (2007 is exception)
        index = 2 if year != 2007 else 1

        # get murder rate
        murder_per_100_k = row.find_all("td")[index].text.strip("\n")

        # update dict, append to list and refresh dictionary
        d.update({"Year":year, "Rate":murder_per_100_k, "MSA":msa})

```

```

        d.update({'murder_per_100_k':murder_per_100_k})
        msa_murders.append(d)
        d = {}

    # create dataframe and drop nan (caused by sub-msa's)
    df = pd.DataFrame(msa_murders)
    df = df.dropna()
    return df

```

```

In [4]: # create dictionary to store dataframes with census data
        fbi_years_dict = {}

        # iterate over years and read the data, storing into dict
        for year in years:
            # read and add to dict
            fbi_years_dict.update({year:fetch_fbi_year_data(year)})
            time.sleep(1)

```

```

In [5]: # take a peak at 2010 data
        fbi_years_dict[2010].head()

```

```

Out[5]:

```

	MSA	murder_per_100_k
0	Abilene, TX	3.1
1	Akron, OH	3.7
2	Albany, GA	8.7
3	Albany-Schenectady-Troy, NY	1.5
4	Albuquerque, NM	5.8

Read Census data

```

In [6]: def read_census_year_data(year):

```

```
'''
Reads Census data by MSA from local files

Inputs:
--year, a year for which data should be read
'''

# parse last two digits of year
year_last_two = str(year)[-2:]

# custom suffix of data (2006 exception)
suffix = '_EST' if year == 2006 else '_1YR'

# set path and name of data files
path = "../data/census/raw/ACS_" + year_last_two + suffix + "_S0201/"
file = "ACS_" + year_last_two + suffix + "_S0201.csv"

# read data
df = pd.read_csv(path + file, header=0, dtype=object)

# add year column to the data
df['year'] = year

# remove suffix in the MSA name
df['GEO.display-label'] = df['GEO.display-label'].apply(lambda x: x[:x.find(" Metro Area")])

# get only subset of columns (avoid MOE columns)
cols = ['year', 'GEO.display-label'] + [c for c in df.columns if c[:3] == 'EST']

# keep only data for all races (total)
df = df[df['POPGROUP.id'] == "001"]

# get filtered columns
df=df[cols]

return df
```

```
In [7]: # create dictionary to store dataframes with census data
census_years_dict = {}

# iterate over years and read the data, storing into dict
for year in years:
    # read and add to dict
    census_years_dict.update({year:read_census_year_data(year)})
```

Prepare 2010 data for EDA

This is done only for 2010 year for EDA, once features subset is decided, further processing will be done for all datasets.

```
In [8]: # get a copy of data
eda_df = census_years_dict[2010].copy()

# rename columns for EDA for 2010 year (TO MAKE NAMES MORE INFORMATIVE AND SHORTER COMPARED TO METAL
col_rename_map_2010 = { 'GEO.display-label':'MSA',
                        'EST_VC11':'total_population',
                        'EST_VC12':'gender_male',
                        'EST_VC13':'gender_female',
                        'EST_VC15':'age_under_5_years',
                        'EST_VC16':'age_5_to_17_years',
                        'EST_VC17':'age_18_to_24_years',
                        'EST_VC18':'age_25_to_34_years',
                        'EST_VC19':'age_35_to_44_years',
                        'EST_VC20':'age_45_to_54_years',
                        'EST_VC21':'age_55_to_64_years',
                        'EST_VC22':'age_65_to_74_years',
                        'EST_VC23':'age_75_years_and_over',
                        'EST_VC25':'age_median_age_(years)',
                        'EST_VC27':'age_18_years_and_over',
                        'EST_VC28':'age_21_years_and_over',
                        'EST_VC29':'age_62_years_and_over',
                        'EST_VC30':'age_65_years_and_over'.
```

```

-----
'EST_VC55': 'population_in_households_householder_or_spouse',
'EST_VC56': 'population_in_households_child',
'EST_VC58': 'population_in_households_nonrelatives',
'EST_VC59': 'population_in_households_nonrelatives_unmarried_partner',
'EST_VC64': 'family_households',
'EST_VC66': 'family_households_with_own_children_under_18_years',
'EST_VC67': 'family_households_married-couple_family',
'EST_VC68': 'family_household_married_couple_family_with_own_children_under_18_years',
'EST_VC69': 'family_households_female_householder_no_husband_present',
'EST_VC70': 'family_households_female_householder_no_husband_present_with_own_children',
'EST_VC71': 'nonfamily_households',
'EST_VC72': 'nonfamily_households_male_householder',
'EST_VC73': 'nonfamily_households_male_householder_living_alone',
'EST_VC74': 'nonfamily_households_male_householder_not_living_alone',
'EST_VC75': 'nonfamily_households_female_householder',
'EST_VC76': 'nonfamily_households_female_householder_living_alone',
'EST_VC77': 'nonfamily_households_female_householder_not_living_alone',
'EST_VC79': 'average_household_size',
'EST_VC80': 'average_family_size',
'EST_VC85': 'now_married_except_separated',
'EST_VC86': 'widowed',
'EST_VC87': 'divorced',
'EST_VC88': 'separated',
'EST_VC89': 'never_married',
'EST_VC108': 'enrolled_nursery_school_or_preschool',
'EST_VC109': 'enrolled_kindergarten',
'EST_VC110': 'enrolled_elementary_school_grades_1_8',
'EST_VC111': 'enrolled_high_school_grades_9_12',
'EST_VC112': 'enrolled_college_or_graduate_school',
'EST_VC124': 'less_than_high_school_diploma',
'EST_VC125': 'high_school_graduate_(includes_equivalency)',
'EST_VC126': 'some_college_or_associates_degree',
'EST_VC127': 'bachelors_degree',
'EST_VC128': 'graduate_or_professional_degree',
'EST_VC130': 'high_school_graduate_or_higher',
'EST_VC133': 'bachelors_degree_or_higher',
'EST_VC142': 'unmarried_portion_of_women_15_to_50_years_who_had_a_birth_in_past_12_months'

```

```

'EST_VC147': 'population_30_years_and_over_living_with_grandchild(ren)',
'EST_VC148': 'population_30_years_and_over_living_with_grandchild(ren)_respon
'EST_VC153': 'civilian_veteran',
'EST_VC158': 'total_civilian_noninst_population_with_a_disability',
'EST_VC161': 'civilian_noninst_population_under_18_years_with_a_disability',
'EST_VC164': 'civilian_noninst_population_18_to_64_years_with_a_disability',
'EST_VC167': 'civilian_noninst_population_65_years_and_older_with_a_disabilit
'EST_VC172': 'residence_year_ago_same_house',
'EST_VC173': 'residence_year_ago_different_house_in_the_us',
'EST_VC174': 'residence_year_ago_different_house_in_the_us_same_county',
'EST_VC175': 'residence_year_ago_different_house_in_the_us_different_county',
'EST_VC176': 'residence_year_ago_different_house_in_the_us_different_county_s
'EST_VC177': 'residence_year_ago_different_house_in_the_us_different_county_d
'EST_VC178': 'residence_year_ago_abroad',
'EST_VC182': 'native',
'EST_VC186': 'foreign_born',
'EST_VC190': 'foreign_born_naturalized_us_citizen',
'EST_VC194': 'foreign_born_not_a_us_citizen',
'EST_VC199': 'born_outside_entered_2000_or_later',
'EST_VC200': 'born_outside_entered_1990_to_1999',
'EST_VC201': 'born_outside_entered_before_1990',
'EST_VC206': 'born_in_europe',
'EST_VC207': 'born_in_asia',
'EST_VC208': 'born_in_africa',
'EST_VC209': 'born_in_oceania',
'EST_VC210': 'born_in_latin_america',
'EST_VC211': 'born_in_northern_america',
'EST_VC216': 'speaking_english_only',
'EST_VC217': 'speaking_language_other_than_english',
'EST_VC218': 'speaking_language_other_than_english_speak_english_less_than_ve
'EST_VC223': 'employment_in_labor_force',
'EST_VC224': 'employment_in_labor_force_civilian_labor_force',
'EST_VC225': 'employment_in_labor_force_civilian_labor_force_employed',
'EST_VC226': 'employment_in_labor_force_civilian_labor_force_unemployed',
'EST_VC227': 'employment_in_labor_force_civilian_labor_force_unemployed_perce
'EST_VC228': 'employment_in_labor_force_armed_forces',
'EST_VC229': 'employment_not_in_labor_force',

```



```
'EST_VC241': 'commuting_car_truck_or_van_drove_alone',
'EST_VC242': 'commuting_car_truck_or_van_carpooled',
'EST_VC243': 'commuting_public_transportation_(excluding_taxicab)',
'EST_VC244': 'commuting_walked',
'EST_VC245': 'commuting_other_means',
'EST_VC246': 'commuting_worked_at_home',
'EST_VC247': 'commuting_mean_travel_time_to_work_(minutes)',
'EST_VC252': 'occupation_management_business_science_and_arts_occupations',
'EST_VC253': 'occupation_service_occupations',
'EST_VC254': 'occupation_sales_and_office_occupations',
'EST_VC255': 'occupation_natural_resources_construction_and_maintenance_occupations',
'EST_VC256': 'occupation_production_transportation_and_material_moving_occupations',
'EST_VC275': 'industry_agriculture_forestry_fishing_and_hunting_and_mining',
'EST_VC276': 'industry_construction',
'EST_VC277': 'industry_manufacturing',
'EST_VC278': 'industry_wholesale_trade',
'EST_VC279': 'industry_retail_trade',
'EST_VC280': 'industry_transportation_and_warehousing_and_utilities',
'EST_VC281': 'industry_information',
'EST_VC282': 'industry_finance_and_insurance_and_real_estate_and_rental_and_utilities',
'EST_VC283': 'industry_professional_scientific_and_management_and_administrative_services',
'EST_VC284': 'industry_educational_services_and_health_care_and_social_assistance_services',
'EST_VC285': 'industry_arts_entertainment_and_recreation_and_accommodation_and_food_services',
'EST_VC286': 'industry_other_services_(except_public_administration)',
'EST_VC287': 'industry_public_administration',
'EST_VC292': 'private_wage_and_salary_workers',
'EST_VC293': 'government_workers',
'EST_VC294': 'self-employed_workers_in_own_not_incorporated_business',
'EST_VC295': 'unpaid_family_workers',
'EST_VC300': 'median_household_income_(dollars)',
'EST_VC302': 'households_with_earnings',
'EST_VC304': 'households_with_social_security_income',
'EST_VC306': 'households_with_supplemental_security_income',
'EST_VC308': 'households_with_cash_public_assistance_income',
'EST_VC310': 'households_with_retirement_income',
'EST_VC312': 'households_with_food_stamp_snap_benefits',
'EST_VC315': 'median_family_income_(dollars)'
```

```

'EST_VC316': 'percentage_married-couple_family',
'EST_VC317': 'median_family_income_(dollars)_married-couple_family',
'EST_VC318': 'percentage_male_householder_no_spouse_present_family',
'EST_VC319': 'median_family_income_(dollars)_male_householder_no_spouse_present_family',
'EST_VC320': 'percentage_female_householder_no_husband_present_family',
'EST_VC321': 'median_family_income_(dollars)_female_householder_no_husband_present_family',
'EST_VC324': 'per_capita_income_(dollars)',
'EST_VC340': 'civilian_noninst_population_with_private_health_insurance',
'EST_VC341': 'civilian_noninst_population_with_public_health_coverage',
'EST_VC342': 'civilian_noninst_population_no_health_insurance_coverage',
'EST_VC345': 'poverty_all_families',
'EST_VC346': 'poverty_all_families_with_related_children_under_18_years',
'EST_VC347': 'poverty_all_families_with_related_children_under_18_years_with_related_children_under_5_years',
'EST_VC348': 'poverty_married-couple_family',
'EST_VC349': 'poverty_married-couple_family_with_related_children_under_18_years',
'EST_VC350': 'poverty_married-couple_family_with_related_children_under_5_years',
'EST_VC351': 'poverty_female_householder_no_husband_present',
'EST_VC352': 'poverty_female_householder_no_husband_present_with_related_children',
'EST_VC353': 'poverty_female_householder_no_husband_present_with_related_children_under_18_years',
'EST_VC355': 'poverty_all_people',
'EST_VC356': 'poverty_under_18_years',
'EST_VC357': 'poverty_related_children_under_18_years',
'EST_VC358': 'poverty_related_children_under_5_years',
'EST_VC359': 'poverty_related_children_5_to_17_years',
'EST_VC360': 'poverty_18_years_and_over',
'EST_VC361': 'poverty_18_to_64_years',
'EST_VC362': 'poverty_65_years_and_over',
'EST_VC363': 'poverty_people_in_families',
'EST_VC364': 'poverty_unrelated_individuals_15_years_and_over',
'EST_VC369': 'owner_occupied_housing_units',
'EST_VC370': 'renter_occupied_housing_units',
'EST_VC372': 'average_household_size_of_owner-occupied_unit',
'EST_VC373': 'average_household_size_of_renter-occupied_unit',
'EST_VC378': 'units_in_structure_1_unit_detached_or_attached',
'EST_VC379': 'units_in_structure_2_to_4_units',
'EST_VC380': 'units_in_structure_5_or_more_units',
'EST_VC381': 'units_in_structure_mobile_home_boat_rv_van_etc',
'EST_VC386': 'housing_units_with_built_2000_or_later'

```

```

    EST_VC386 : housing_unit_built_2000_or_later ,
'EST_VC387': 'housing_unit_built_1990_to_1999',
'EST_VC388': 'housing_unit_built_1980_to_1989',
'EST_VC389': 'housing_unit_built_1960_to_1979',
'EST_VC390': 'housing_unit_built_1940_to_1959',
'EST_VC391': 'housing_unit_built_1939_or_earlier',
'EST_VC396': 'vehicles_per_housing_unit_none',
'EST_VC397': 'vehicles_per_housing_unit_1_or_more',
'EST_VC402': 'house_heating_fuel_gas',
'EST_VC403': 'house_heating_fuel_electricity',
'EST_VC404': 'house_heating_fuel_all_other_fuels',
'EST_VC405': 'house_heating_fuel_no_fuel_used',
'EST_VC409': 'occupied_housing_units',
'EST_VC410': 'no_telephone_service_available',
'EST_VC411': '1_01_or_more_occupants_per_room',
'EST_VC416': 'monthly_owner_costs_as_percentage_of_household_income_less_than_30_percent',
'EST_VC417': 'monthly_owner_costs_as_percentage_of_household_income_30_percent_or_more',
'EST_VC422': 'house_median_value_(dollars)',
'EST_VC423': 'house_median_selected_monthly_owner_costs_with_a_mortgage_(dollars)',
'EST_VC424': 'house_median_selected_monthly_owner_costs_without_a_mortgage_(dollars)',
'EST_VC429': 'gross_rent_as_percentage_of_household_income_less_than_30_percent',
'EST_VC430': 'gross_rent_as_percentage_of_household_income_30_percent_or_more',
'EST_VC435': 'median_gross_rent_(dollars)'}

# rename the columns
eda_df = eda_df.rename(columns=col_rename_map_2010)

# get list of columns to retain
cols_to_keep = [c for c in eda_df.columns if c[:3] != 'EST']

# update columns
eda_df = eda_df[cols_to_keep]

# take a peak at dataframe
eda_df.head()

```

Out[8]:

	year	MSA	total_population	gender_male	gender_female	age_under_5_years	age_5_to_17_years	age_18_to_24_years	age
1	2010	Akron, OH	702951	48.6	51.4	5.6	16.7	10.6	
2	2010	Albany-Schenectady-Troy, NY	870832	48.9	51.1	5.4	15.9	11.1	
3	2010	Albuquerque, NM	892014	49.0	51.0	6.8	17.6	9.8	
4	2010	Allentown-Bethlehem-Easton, PA-NJ	822141	48.8	51.2	5.7	17.1	8.8	
5	2010	Atlanta-Sandy Springs-Marietta, GA	5288302	48.7	51.3	7.2	19.3	9.2	

5 rows × 190 columns

```
In [9]: # join FBI and Census data
# ATTENTION some MSA's data will be lost due to unmatched, but more than 90% of census MSA match to FBI
eda_df = pd.merge(eda_df, fbi_years_dict[2010], on=['MSA'])
```

```
In [10]: # export data to CSV and pickle
eda_df.to_csv("../data/merged/eda_2010.csv", sep=',', index=False)
eda_df.to_pickle("../data/merged/eda_2010.pkl")
```

Prepare all data for modeling

```
In [11]: # create dictionary of selected feature names (easier to rename if needed)
census_features_dict = {
    'feature_1': 'now_married_except_separated',
```

```
'feature_2':'less_than_high_school_diploma',
'feature_3':'unmarried_portion_of_women_15_to_50_years_who_had_a_birth_in_past_12_months',
'feature_4':'households_with_food_stamp_snap_benefits',
'feature_5':'percentage_married-couple_family',
'feature_6':'percentage_female_householder_no_husband_present_family',
'feature_7':'poverty_all_people',
'feature_8':'house_median_value_(dollars)'
}

# create dictionary to store mapping dictionaries of feature codes for each year, and update the dict
cols_mapping_dicts = {}

cols_mapping_dicts[2006] = {
    'GEO.display-label':'MSA',
    'EST_VC60':census_features_dict['feature_1'],
    'EST_VC92':census_features_dict['feature_2'],
    'EST_VC107':census_features_dict['feature_3'],
    'EST_VC242':census_features_dict['feature_4'],
    'EST_VC245':census_features_dict['feature_5'],
    'EST_VC249':census_features_dict['feature_6'],
    'EST_VC272':census_features_dict['feature_7'],
    'EST_VC316':census_features_dict['feature_8']}

cols_mapping_dicts[2007] = {
    'GEO.display-label':'MSA',
    'EST_VC65':census_features_dict['feature_1'],
    'EST_VC97':census_features_dict['feature_2'],
    'EST_VC112':census_features_dict['feature_3'],
    'EST_VC247':census_features_dict['feature_4'],
    'EST_VC250':census_features_dict['feature_5'],
    'EST_VC254':census_features_dict['feature_6'],
    'EST_VC277':census_features_dict['feature_7'],
    'EST_VC321':census_features_dict['feature_8']}

cols_mapping_dicts[2008] = {
    'GEO.display-label':'MSA',
    'EST_VC66':census_features_dict['feature_1'],
```

```

        'EST_VC98':census_features_dict['feature_2'],
        'EST_VC113':census_features_dict['feature_3'],
        'EST_VC249':census_features_dict['feature_4'],
        'EST_VC252':census_features_dict['feature_5'],
        'EST_VC256':census_features_dict['feature_6'],
        'EST_VC279':census_features_dict['feature_7'],
        'EST_VC329':census_features_dict['feature_8']}]

cols_mapping_dicts[2009] = {
    'GEO.display-label':'MSA',
    'EST_VC66':census_features_dict['feature_1'],
    'EST_VC98':census_features_dict['feature_2'],
    'EST_VC113':census_features_dict['feature_3'],
    'EST_VC249':census_features_dict['feature_4'],
    'EST_VC252':census_features_dict['feature_5'],
    'EST_VC256':census_features_dict['feature_6'],
    'EST_VC284':census_features_dict['feature_7'],
    'EST_VC334':census_features_dict['feature_8']}]

cols_mapping_dicts[2010] = {
    'GEO.display-label':'MSA',
    'EST_VC85':census_features_dict['feature_1'],
    'EST_VC124':census_features_dict['feature_2'],
    'EST_VC142':census_features_dict['feature_3'],
    'EST_VC312':census_features_dict['feature_4'],
    'EST_VC316':census_features_dict['feature_5'],
    'EST_VC320':census_features_dict['feature_6'],
    'EST_VC355':census_features_dict['feature_7'],
    'EST_VC422':census_features_dict['feature_8']}]

# here the mappings for the years are the same
cols_mapping_dicts[2011] = cols_mapping_dicts[2010]
cols_mapping_dicts[2012] = cols_mapping_dicts[2011]

cols_mapping_dicts[2013] = {
    'GEO.display-label':'MSA',
    'EST_VC93':census_features_dict['feature_1'],
    'EST_VC125':census_features_dict['feature_2'],

```

```
    'EST_VC135':census_features_dict['feature_2'],
    'EST_VC154':census_features_dict['feature_3'],
    'EST_VC332':census_features_dict['feature_4'],
    'EST_VC336':census_features_dict['feature_5'],
    'EST_VC340':census_features_dict['feature_6'],
    'EST_VC376':census_features_dict['feature_7'],
    'EST_VC444':census_features_dict['feature_8']}]

# here the mappings for the years are the same
cols_mapping_dicts[2014] = cols_mapping_dicts[2013]

cols_mapping_dicts[2015] = {
    'GEO.display-label':'MSA',
    'EST_VC93':census_features_dict['feature_1'],
    'EST_VC135':census_features_dict['feature_2'],
    'EST_VC154':census_features_dict['feature_3'],
    'EST_VC332':census_features_dict['feature_4'],
    'EST_VC336':census_features_dict['feature_5'],
    'EST_VC340':census_features_dict['feature_6'],
    'EST_VC376':census_features_dict['feature_7'],
    'EST_VC445':census_features_dict['feature_8']}]

# here the mappings for the years are the same
cols_mapping_dicts[2016] = cols_mapping_dicts[2015]
```

```
In [12]: def get_selected_features(df, mapping_dict):  
    '''  
    Selects features specified, maps the names, and returns the df with only those features  
  
    Inputs:  
    --df, a dataframe for which selection is made  
    --mapping_dict, dictionary containing subset of features with mappings  
    '''  
    # get a copy of data  
    df = df.copy()  
  
    # rename the columns and get only those  
    df = df.rename(columns=mapping_dict)  
    df = df[['year']+list(mapping_dict.values())]  
  
    return df
```

```
In [13]: # create df to store all data  
all_df = pd.DataFrame()  
  
# repeat for each year adding joined data into one dataframe  
for year in years:  
    # get census processed data and merge to fbi data  
    year_df = pd.merge(get_selected_features(census_years_dict[year], cols_mapping_dicts[year]),  
                       fbi_years_dict[year],  
                       on=['MSA'])  
  
    # add data for this year into the combined dataframe  
    all_df = pd.concat([all_df, year_df])  
  
# reset index  
all_df = all_df.reset_index(drop=True)  
  
# set datatypes  
all_df['year'] = all_df['year'].astype(int)
```



```

all_df['murder_per_100_k'] = all_df['murder_per_100_k'].astype(float)
all_df[census_features_dict['feature_1']] = all_df[census_features_dict['feature_1']].astype(float)
all_df[census_features_dict['feature_2']] = all_df[census_features_dict['feature_2']].astype(float)
all_df[census_features_dict['feature_3']] = all_df[census_features_dict['feature_3']].astype(float)
all_df[census_features_dict['feature_4']] = all_df[census_features_dict['feature_4']].astype(float)
all_df[census_features_dict['feature_5']] = all_df[census_features_dict['feature_5']].astype(float)
all_df[census_features_dict['feature_6']] = all_df[census_features_dict['feature_6']].astype(float)
all_df[census_features_dict['feature_7']] = all_df[census_features_dict['feature_7']].astype(float)
all_df[census_features_dict['feature_8']] = all_df[census_features_dict['feature_8']].astype(int)

# take a look at dataframe
all_df.head()

```

Out[13]:

	year	MSA	now_married_except_separated	less_than_high_school_diploma	unmarried_portion_of_women_15_to_50_years_who
0	2006	Atlanta-Sandy Springs-Marietta, GA	49.2		14.2
1	2006	Austin-Round Rock, TX	48.7		13.7
2	2006	Baltimore-Towson, MD	47.2		14.0
3	2006	Birmingham-Hoover, AL	50.9		15.8
4	2006	Buffalo-Niagara Falls, NY	47.1		12.9

In [14]: *# create dictionary to rename some of the unmatched across the years MSAs*
some MSA were potentially resised, but we've chosen to create approximate grouping
which will be used for analysis including MSA, but won't affect model with census features only

```
msa_orig_map_to_corr = {'Atlanta-Sandy Springs-Roswell, GA':'Atlanta-Sandy Springs-Marietta, GA',
'Austin-Round Rock-San Marcos, TX':'Austin-Round Rock, TX',
'Bakersfield-Delano, CA':'Bakersfield, CA',
'Baltimore-Towson, MD':'Baltimore-Columbia-Towson, MD',
'Boise City, ID':'Boise City-Nampa, ID',
'Boston-Cambridge-Newton, MA-NH':'Boston-Cambridge-Quincy, MA-NH',
'Buffalo-Niagara Falls, NY':'Buffalo-Cheektowaga-Niagara Falls, NY',
'Charleston-North Charleston, SC':'Charleston-North Charleston-Summerville',
'Charlotte-Gastonia-Concord, NC-SC':'Charlotte-Concord-Gastonia, NC-SC',
'Charlotte-Gastonia-Rock Hill, NC-SC':'Charlotte-Concord-Gastonia, NC-SC',
'Chicago-Naperville-Elgin, IL-IN-WI':'Chicago-Joliet-Naperville, IL-IN-WI',
'Cincinnati, OH-KY-IN':'Cincinnati-Middletown, OH-KY-IN',
'Cleveland-Elyria, OH':'Cleveland-Elyria-Mentor, OH',
'Denver-Aurora-Broomfield, CO':'Denver-Aurora, CO',
'Denver-Aurora-Lakewood, CO':'Denver-Aurora, CO',
'Detroit-Warren-Livonia, MI':'Detroit-Warren-Dearborn, MI',
'Greenville-Anderson-Mauldin, SC':'Greenville-Mauldin-Easley, SC',
'Urban Honolulu, HI':'Honolulu, HI',
'Houston-The Woodlands-Sugar Land, TX':'Houston-Sugar Land-Baytown, TX',
'Indianapolis-Carmel, IN':'Indianapolis-Carmel-Anderson, IN',
'Las Vegas-Paradise, NV':'Las Vegas-Henderson-Paradise, NV',
'Los Angeles-Long Beach-Santa Ana, CA':'Los Angeles-Long Beach-Anaheim, CA',
'Miami-Fort Lauderdale-Pompano Beach, FL':'Miami-Fort Lauderdale-Miami Beach',
'Miami-Fort Lauderdale-West Palm Beach, FL':'Miami-Fort Lauderdale-Miami Beach',
'New Orleans-Metairie, LA':'New Orleans-Metairie-Kenner, LA',
'New York-Newark-Jersey City, NY-NJ-PA':'New York-Northern New Jersey-Long Island',
'North Port-Bradenton-Sarasota, FL':'North Port-Sarasota-Bradenton, FL',
'Orlando-Kissimmee, FL':'Orlando-Kissimmee-Sanford, FL',
'Phoenix-Mesa-Glendale, AZ':'Phoenix-Mesa-Scottsdale, AZ',
'Portland-South Portland, ME':'Portland-South Portland-Biddeford, ME',
'Portland-Vancouver-Hillsboro, OR-WA':'Portland-Vancouver-Beaverton, OR-WA',
'Providence-Warwick, RI-MA':'Providence-New Bedford-Fall River, RI-MA',
'Raleigh, NC':'Raleigh-Cary, NC',
'San Antonio, TX':'San Antonio-New Braunfels, TX',
'San Diego-Carlsbad, CA':'San Diego-Carlsbad-San Marcos, CA',
'San Francisco-Oakland-Hayward, CA':'San Francisco-Oakland-Fremont, CA',
'Stockton, CA':'Stockton-Lodi, CA'.
```

```

-----, ---, -----, ---,
'Worcester, MA': 'Worcester, MA-CT'}

# create dictionary to also create column with abbreviated MSAs for easier modeling using hot encoding
msa_corr_map_to_abbr = {'Akron, OH': 'AKRON_OH',
                        'Albany-Schenectady-Troy, NY': 'ALBANY_NY',
                        'Albuquerque, NM': 'ALBUQUERQUE_NM',
                        'Allentown-Bethlehem-Easton, PA-NJ': 'ALLENTOWN_PA',
                        'Atlanta-Sandy Springs-Marietta, GA': 'ATLANTA_GA',
                        'Augusta-Richmond County, GA-SC': 'AUGUSTA_GA',
                        'Austin-Round Rock, TX': 'AUSTIN_TX',
                        'Bakersfield, CA': 'BAKERSFIELD_CA',
                        'Baltimore-Columbia-Towson, MD': 'BALTIMORE_MD',
                        'Baton Rouge, LA': 'BATON_ROUGE_LA',
                        'Birmingham-Hoover, AL': 'BIRMINGHAM_AL',
                        'Boise City-Nampa, ID': 'BOISE_ID',
                        'Boston-Cambridge-Quincy, MA-NH': 'BOSTON_MA',
                        'Bradenton-Sarasota-Venice, FL': 'BRADENTON_FL',
                        'Bridgeport-Stamford-Norwalk, CT': 'BRIDGEPORT_CT',
                        'Buffalo-Cheektowaga-Niagara Falls, NY': 'BUFFALO_NY',
                        'Cape Coral-Fort Myers, FL': 'CAPE_CORAL_FL',
                        'Charleston-North Charleston-Summerville, SC': 'CHARLESTON_SC',
                        'Charlotte-Concord-Gastonia, NC-SC': 'CHARLOTTE_NC',
                        'Chattanooga, TN-GA': 'CHATANOOGA_TN',
                        'Chicago-Joliet-Naperville, IL-IN-WI': 'CHICAGO_IL',
                        'Cincinnati-Middletown, OH-KY-IN': 'CINCINNATI_OH',
                        'Cleveland-Elyria-Mentor, OH': 'CLEVELAND_OH',
                        'Colorado Springs, CO': 'COLORADO_SPRINGS_CO',
                        'Columbia, SC': 'COLUMBIA_SC',
                        'Columbus, OH': 'COLUMBUS_OH',
                        'Dallas-Fort Worth-Arlington, TX': 'DALLAS_TX',
                        'Dayton, OH': 'DAYTON_OH',
                        'Deltona-Daytona Beach-Ormond Beach, FL': 'DELTONA_FL',
                        'Denver-Aurora, CO': 'DENVER_CO',
                        'Des Moines-West Des Moines, IA': 'DES_MOINES_IA',
                        'Detroit-Warren-Dearborn, MI': 'DETROIT_MI',
                        'Durham-Chapel Hill, NC': 'DURHAM_NC',
                        'El Paso, TX': 'EL PASO_TX',

```

```
'Fresno, CA': 'FRESNO_CA',  
'Grand Rapids-Wyoming, MI': 'GRAND_RAPIDS_MI',  
'Greensboro-High Point, NC': 'GREENSBORO_NC',  
'Greenville-Mauldin-Easley, SC': 'GREENVILLE_SC',  
'Harrisburg-Carlisle, PA': 'HARRISBURG_PA',  
'Hartford-West Hartford-East Hartford, CT': 'HARTFORD_CT',  
'Honolulu, HI': 'HONOLULU_HI',  
'Houston-Sugar Land-Baytown, TX': 'HOUSTON_TX',  
'Indianapolis-Carmel-Anderson, IN': 'INDIANAPOLIS_IN',  
'Jackson, MS': 'JACKSON_MS',  
'Jacksonville, FL': 'JACKSONVILLE_FL',  
'Kansas City, MO-KS': 'KANSAS_CITY_MO',  
'Knoxville, TN': 'KNOXVILLE_TN',  
'Lakeland-Winter Haven, FL': 'LAKE LAND_FL',  
'Lancaster, PA': 'LANCASTER_PA',  
'Las Vegas-Henderson-Paradise, NV': 'LAS_VEGAS_NV',  
'Lexington-Fayette, KY': 'LEXINGTON_KY',  
'Little Rock-North Little Rock-Conway, AR': 'LITTLE_ROCK_AR',  
'Los Angeles-Long Beach-Anaheim, CA': 'LOS_ANGELES_CA',  
'Louisville-Jefferson County, KY-IN': 'LOUISVILLE_KY',  
'Louisville/Jefferson County, KY-IN': 'LOUISVILLE_KY',  
'Madison, WI': 'MADISON_WI',  
'McAllen-Edinburg-Mission, TX': 'MCALLEN_TX',  
'Memphis, TN-MS-AR': 'MEMPHIS_TN',  
'Miami-Fort Lauderdale-Miami Beach, FL': 'MIAMI_FL',  
'Milwaukee-Waukesha-West Allis, WI': 'MILWAUKEE_WI',  
'Minneapolis-St. Paul-Bloomington, MN-WI': 'MINNEAPOLIS_MN',  
'Modesto, CA': 'MODESTO_CA',  
'Nashville-Davidson--Murfreeseboro--Franklin, TN': 'NASHVILLE_TN',  
'New Haven-Milford, CT': 'NEW_HAVEN_CT',  
'New Orleans-Metairie-Kenner, LA': 'NEW_ORLEANS_LA',  
'New York-Northern New Jersey-Long Island, NY-NJ-PA': 'NEW_YORK_NY',  
'North Port-Sarasota-Bradenton, FL': 'NORTH_PORT_FL',  
'Ogden-Clearfield, UT': 'OGDEN_UT',  
'Oklahoma City, OK': 'OKLAHOMA_CITY_OK',  
'Omaha-Council Bluffs, NE-IA': 'OMAHA_NE',  
'Orlando-Kissimmee-Sanford, FL': 'ORLANDO_FL',
```

```
'Oxnard-Thousand Oaks-Ventura, CA': 'OXNARD_CA',  
'Palm Bay-Melbourne-Titusville, FL': 'PALM_BAY_FL',  
'Philadelphia-Camden-Wilmington, PA-NJ-DE-MD': 'PHILADELPHIA_PA',  
'Phoenix-Mesa-Scottsdale, AZ': 'PHOENIX_AZ',  
'Pittsburgh, PA': 'PITTSBURGH_PA',  
'Portland-South Portland-Biddeford, ME': 'PORTLAND_ME',  
'Portland-Vancouver-Beaverton, OR-WA': 'PORTLAND_OR',  
'Poughkeepsie-Newburgh-Middletown, NY': 'POUGHKEEPSIE_NY',  
'Providence-New Bedford-Fall River, RI-MA': 'PROVIDENCE_RI',  
'Provo-Orem, UT': 'PROVO_UT',  
'Raleigh-Cary, NC': 'RALEIGH_NC',  
'Richmond, VA': 'RICHMOND_VA',  
'Riverside-San Bernardino-Ontario, CA': 'RIVERSIDE_CA',  
'Rochester, NY': 'ROCHESTER_NY',  
'Salt Lake City, UT': 'SALT_LAKE_CITY_UT',  
'San Antonio-New Braunfels, TX': 'SAN_ANTONIO_TX',  
'San Diego-Carlsbad-San Marcos, CA': 'SAN_DIEGO_CA',  
'San Francisco-Oakland-Fremont, CA': 'SAN_FRANCISCO_CA',  
'San Jose-Sunnyvale-Santa Clara, CA': 'SAN_JOSE_CA',  
'Santa Rosa, CA': 'SANTA_ROSA_CA',  
'Seattle-Tacoma-Bellevue, WA': 'SEATTLE_WA',  
'Spokane-Spokane Valley, WA': 'SPOKANE_WA',  
'Springfield, MA': 'SPRINGFIELD_MA',  
'St. Louis, MO-IL': 'ST_LOUIS_MO',  
'Stockton-Lodi, CA': 'STOCKTON_CA',  
'Syracuse, NY': 'SYRACUSE_NY',  
'Tampa-St. Petersburg-Clearwater, FL': 'TAMPA_FL',  
'Toledo, OH': 'TOLEDO_OH',  
'Tucson, AZ': 'TUCSON_AZ',  
'Tulsa, OK': 'TULSA_OK',  
'Virginia Beach-Norfolk-Newport News, VA-NC': 'VIRGINIA_BEACH_NC',  
'Washington-Arlington-Alexandria, DC-VA-MD-WV': 'WASHINGTON_DC',  
'Wichita, KS': 'WICHITA_KS',  
'Winston-Salem, NC': 'WINSTON_NC',  
'Worcester, MA-CT': 'WORCESTER_MA',  
'Youngstown-Warren-Boardman, OH-PA': 'YOUNGSTOWN_OH'}
```

```

In [15]: #rename msa column to msa_orig
all_df = all_df.rename(columns={'MSA':'MSA_orig'})

# create new column with corrected names
all_df['MSA_corr'] = all_df['MSA_orig']\
    .apply(lambda x :msa_orig_map_to_corr.get(x) if msa_orig_map_to_corr.get(x) is not None else x)

# creat additional column with abbreviated names
all_df['MSA_abbr'] = all_df['MSA_corr'].apply(lambda x : msa_corr_map_to_abbr.get(x))

# set columns list (in desired order)
columns = ['MSA_orig', 'MSA_corr', 'MSA_abbr', 'year',
            'now_married_except_separated',
            'less_than_high_school_diploma',
            'unmarried_portion_of_women_15_to_50_years_who_had_a_birth_in_past_12_months',
            'households_with_food_stamp_snap_benefits',
            'percentage_married-couple_family',
            'percentage_female_householder_no_husband_present_family',
            'poverty_all_people', 'house_median_value_(dollars)',
            'murder_per_100_k']

# reorder columns
all_df = all_df[columns]

# take a look at df
all_df.head()

```

Out[15]:

	MSA_orig	MSA_corr	MSA_abbr	year	now_married_except_separated	less_than_high_school_diploma	unmarried_porti
0	Atlanta-Sandy Springs-Marietta, GA	Atlanta-Sandy Springs-Marietta, GA	ATLANTA_GA	2006	49.2	14.2	
1	Austin-Round Rock, TX	Austin-Round Rock, TX	AUSTIN_TX	2006	48.7	13.7	

2	Baltimore-Towson, MD	Baltimore-Columbia-Towson, MD	BALTIMORE_MD	2006	47.2	14.0
3	Birmingham-Hoover, AL	Birmingham-Hoover, AL	BIRMINGHAM_AL	2006	50.9	15.8
4	Buffalo-Niagara Falls, NY	Buffalo-Cheektowaga-Niagara Falls, NY	BUFFALO_NY	2006	47.1	12.9

```
In [16]: # export dataframe into csv and pickle
all_df.to_csv("../data/merged/all_data_2006_to_2016.csv", sep=',', index=False)
all_df.to_pickle("../data/merged/all_data_2006_to_2016.pkl")
```