United States Census Bureau

Source: https://www.census.gov

Course: Big Data Lab 1 (W.MSCIDS_BDL03_1.H21)

Author: Alexandra Alinka Zimmermann

Date: 01.11.2021

United States Census Project

For the course *Big Data Lab 1* we were asked to work on a project with the aim of applying and deepening our knowledge on MongoDB. I decided to work on a dataset from the United States Census Bureau.

Originally, the data was fetched from the United States Census Bureau (Data source). It was then compiled, saved into a JSON file (JSON file source) and stored online, where I requested the access to the file's data. For additional information, an explanation of the questions asked in the census can be found by first accessing the link below, under "Data source & Documentation" and then by clicking on (i) next to the row of interest.

JSON file source: County demographics JSON file

Data source & Documentation United States Census Bureau

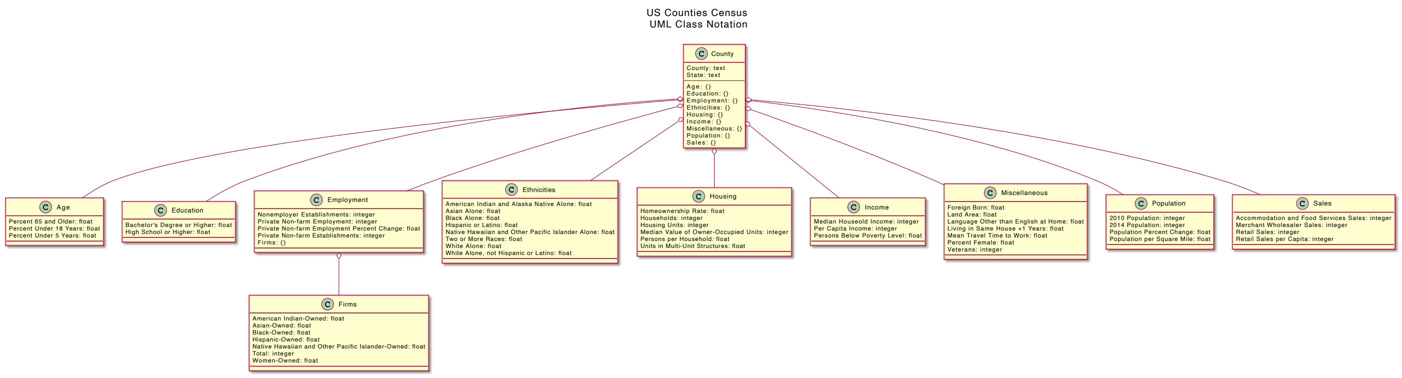
Table of Contents

1 Overview

This census was conducted by United States Census Bureau, part of the U.S. Department of Commerce, between 2007 to 2014 at the County level. There are 3,143 Counties, for which different parameters are recorded, such as:

- the State they belong to
- the size of the population
- the age and gender distribution
- the relative size of different ethnic groups
- other population characteristics
- some details about households
- the education level
- some economics details
- some income and poverty indicators
- some business details

Below, the Unified Modeling language (UML) Class diagram is presented. It displays the structure of the latest version of my dataset. Indeed, in step 3.4.1, I dropped two fields from all documents in the collection.


US Counties Census
UML Class Notation
C County
County: text
State: text
Age: {}
Education: {}
Employment: {}
Ethnicities: {}
Housing: {}
Income: {}
Miscellaneous: {}
Population: {}
Sales: {}
C Age
Percent 65 and Older: float
Percent Under 18 Years: float
Percent Under 5 Years: float
C Education
Bachelor's Degree or Higher: float
High School or Higher: float
C Employment
Nonemployer Establishments: integer
Private Non-farm Employment: integer
Private Non-farm Employment Percent Change: float
Private Non-farm Establishments: integer
Firms: {}
C Firms
American Indian-Owned: float
Asian-Owned: float
Black-Owned: float
Hispanic-Owned: float
Native Hawaiian and Other Pacific Islander-Owned: float
Total: integer
Women-Owned: float
C Ethnicities
American Indian and Alaska Native Alone: float
Asian Alone: float
Black Alone: float
Hispanic or Latino: float
Native Hawaiian and Other Pacific Islander Alone: float
Two or More Races: float
White Alone: float
White Alone, not Hispanic or Latino: float
C Housing
Homeownership Rate: float
Households: integer
Housing Units: integer
Median Value of Owner-Occupied Units: integer
Persons per Household: float
Units in Multi-Unit Structures: float
C Income
Median Houseold Income: integer
Per Capita Income: integer
Persons Below Poverty Level: float
C Miscellaneous
Foreign Born: float
Land Area: float
Language Other than English at Home: float
Living in Same House +1 Years: float
Mean Travel Time to Work: float
Percent Female: float
Veterans: integer
C Population
2010 Population: integer
2014 Population: integer
Population Percent Change: float
Population per Square Mile: float
C Sales
Accommodation and Food Services Sales: integer
Merchant Wholesaler Sales: integer
Retail Sales: integer
Retail Sales per Capita: integer

A census is essential as it helps to better understand a given society at a particular time. If the statistics are reliable enough, it provides a close representation of the population's current situation. It uncovers possible disparities and also highlights some needs that could be fulfilled for the benefit of the population.

2 Requirements & Configuration

```
! pip3 list | findstr "pymongo dnspython pandas"
```

2.1 Import packages

```
import pymongo
from pprint import pprint
import pandas as pd
import requests
import json
import numpy as np

# Import the necessary packages to download the online file
import requests
import urllib

# To visualise correlation
import seaborn as sns
```

2.2 Pandas configuration

```
# Pandas configuration
pd.set_option('precision', 2)
pd.set_option('max_rows', 51) # there are 51 different States in the United States
# Side note: Washington, District of Columbia is not really a State
# but rather a Federal District but I will not make any difference in this project.

pd.set_option('max_colwidth', 50)
# pd.describe_option('max_rows')
# pd.describe_option('precision')
# pd.describe_option('max_colwidth')
```

2.3 Access to the data

```
# Data URL
url = "https://corgis-edu.github.io/corgis/datasets/json/county_demographics/county_demographics.json"
```

2.4 Database details

```
CNX_STR = "mongodb+srv://Alinka:MongoDBAlinka96@bigdatalab1.whieg.mongodb.net"
DB_NAME = "projectBDL1"
COLL_NAME = "county_demographics"
```

```
# Connection to MongoDB
client = pymongo.MongoClient(CNX_STR)
db = client[DB_NAME]
county = db[COLL_NAME]
```

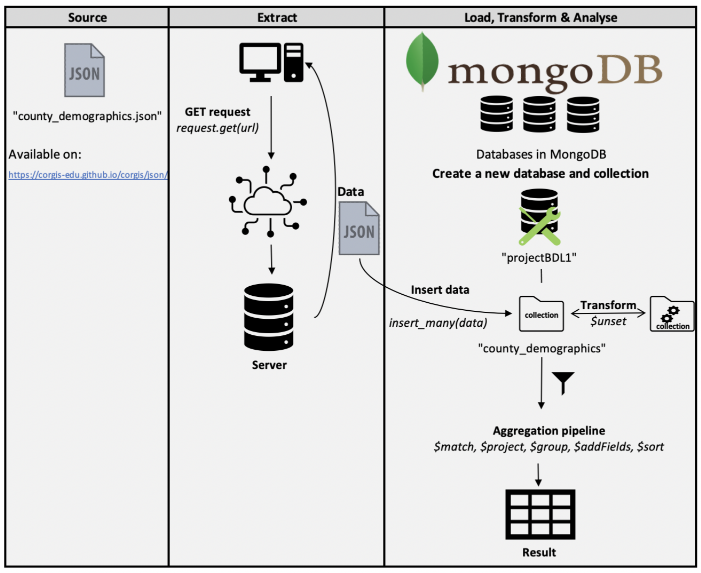
```
# Databases available on MongoDB and the space they occupy
dbs = pd.DataFrame(client.list_databases()) # Not printed here
```

3 ETL

The dataflow can be visualised in the image below.

In order to use the data, I first made a request from the server and saved the JSON file in a variable.

On MongoDB, I created a new database called "projectBDL1" and a new collection "county_demographics". I imported the JSON file, meaning all the documents that it contained, into the collection. Only for learning purposes, I transformed all the documents by removing two fields. Finally, I applied various aggregation pipelines on the documents and obtained the desired results.



3.1 Remove all existing documents

To reset the "county_demographics" collection.

```
county.drop()
county.count_documents({})
```

```
0
```

3.2 Fetch data

```
data = requests.get(url).json()

print("The type of data is:", type(data))

# Each element of the list is a dictionary, one for each County
print("The type of data[0] is:", type(data[0]))
```

```
The type of data is: <class 'list'>
The type of data[0] is: <class 'dict'>
```

3.3 Insert into MongoDB

```
# Insert the dictionaries of the list 'data' into MongoDB collection 'county_demographics'
county.insert_many(data)
```

```
<pymongo.results.InsertManyResult at 0x7fa6862cf3c0>
```

```
# Count the number of documents inserted
county.count_documents({})
```

```
3143
```

```
# Visualise one document of the collection county_demographics
county.find_one({})
```

```
{'_id': ObjectId('617fbd61b61a9175f49cf8e5'),
 'County': 'Autauga County',
 'State': 'AL',
 'Age': {'Percent 65 and Older': 13.8,
  'Percent Under 18 Years': 25.2,
  'Percent Under 5 Years': 6.0},
 'Education': {"Bachelor's Degree or Higher": 20.9,
  'High School or Higher': 85.6},
 'Employment': {'Nonemployer Establishments': 2947,
  'Private Non-farm Employment': 10120,
  'Private Non-farm Employment Percent Change': 2.1,
  'Private Non-farm Establishments': 817,
  'Firms': {'American Indian-Owned': 0.0,
   'Asian-Owned': 1.3,
   'Black-Owned': 15.2,
   'Hispanic-Owned': 0.7,
   'Native Hawaiian and Other Pacific Islander-Owned': 0.0,
   'Total': 4067,
   'Women-Owned': 31.7}},
 'Ethnicities': {'American Indian and Alaska Native Alone': 0.5,
  'Asian Alone': 1.1,
  'Black Alone': 18.7,
  'Hispanic or Latino': 2.7,
  'Native Hawaiian and Other Pacific Islander Alone': 0.1,
  'Two or More Races': 1.8,
  'White Alone': 77.9,
  'White Alone, not Hispanic or Latino': 75.6},
 'Housing': {'Homeownership Rate': 76.8,
  'Households': 20071,
  'Housing Units': 22751,
  'Median Value of Owner-Occupied Units': 136200,
  'Persons per Household': 2.71,
  'Units in Multi-Unit Structures': 8.3},
 'Income': {'Median Houseold Income': 53682,
  'Per Capita Income': 24571,
  'Persons Below Poverty Level': 12.1},
 'Miscellaneous': {'Building Permits': 131,
  'Foreign Born': 1.6,
  'Land Area': 594.44,
  'Language Other than English at Home': 3.5,
  'Living in Same House +1 Years': 85.0,
  'Manufacturers Shipments': 0,
  'Mean Travel Time to Work': 26.2,
  'Percent Female': 51.4,
  'Veterans': 5922},
 'Population': {'2010 Population': 54571,
  '2014 Population': 55395,
  'Population Percent Change': 1.5,
  'Population per Square Mile': 91.8},
 'Sales': {'Accommodation and Food Services Sales': 881,
  'Merchant Wholesaler Sales': 0,
  'Retail Sales': 5981,
  'Retail Sales per Capita': 12003}}
```

3.4 Transform

3.4.1 Drop fields

Only for learning purposes, I dropped 'Building Permits' and 'Manufacturers Shipments' from the field 'Miscellaneous'.

```
# Print a list of the nested fields available in Miscellaneous (before cleaning) and its length
print('Original length:',
      len(county.find_one({},{"Miscellaneous":1,"_id":0})["Miscellaneous"].keys())) # length

pprint(list(county.find_one({},{"Miscellaneous":1,"_id":0})["Miscellaneous"].keys())) # nested fields
```

```
Original length: 9
['Building Permits',
 'Foreign Born',
 'Land Area',
 'Language Other than English at Home',
 'Living in Same House +1 Years',
 'Manufacturers Shipments',
 'Mean Travel Time to Work',
 'Percent Female',
 'Veterans']
```

```
# Drop the fields "Building Permits" and "Manufacturers Shipments"
try:
    county.update_many({}, {'$unset': {'Miscellaneous.Building Permits':1,
                                       'Miscellaneous.Manufacturers Shipments':1}})
    print('Nested fields (Building Permits, Manufacturers) were removed')
except Exception:
    print('Error found') # if the drop does not work, I would get a warning
```

```
Nested fields (Building Permits, Manufacturers) were removed
```

```
# Print a list of the nested fields available in Miscellaneous (after cleaning) and its length
print('Final length:',
      len(county.find_one({},{"Miscellaneous":1,"_id":0})["Miscellaneous"].keys()))

pprint(list(county.find_one({},{"Miscellaneous":1,"_id":0})["Miscellaneous"].keys()))
```

```
Final length: 7
['Foreign Born',
 'Land Area',
 'Language Other than English at Home',
 'Living in Same House +1 Years',
 'Mean Travel Time to Work',
 'Percent Female',
 'Veterans']
```

3.4.2 Find duplicates

First, I needed to verify whether some places had the same County name across different States.

```
count_county = county.aggregate([
    # Group by County name and count the number of times the same name is repeated
    {"$group" : { "_id": "$County", "count": { "$sum": 1 }}},

    # Keep only the County names that are repeated more than once
    {"$match": {"count" : {"$gt": 1}}},

    # Display the results by keeping only the County names
    # and the number of times they appear in the dataset
    {"$project": {
        "_id": 0,
        "county": "$_id", # Rename "_id" to "county"
        "count": "$count"}},

    # Sort by the number of times they appear, in ascending order
    {"$sort": {"count":1}}
])

pd.DataFrame(count_county)
```

	county	count
0	Hillsborough County	2
1	Seneca County	2
2	Champaign County	2
3	Scotland County	2
4	Yuma County	2
...	...	...
419	Lincoln County	23
420	Jackson County	23
421	Franklin County	24
422	Jefferson County	25
423	Washington County	30

424 rows × 2 columns

It seemed that "Lincoln County" name was repeated 23 times in this collection.

I displayed below in which States this County name appeared.

```
pd.DataFrame(county.aggregate([
    # Look for the County name Lincoln, using regex, without being case sensitive
    {'$match': {'County':
                {"$regex" : "lincoln",  "$options": "i"}}}, # option = i, to have case insensitive

    # Keep only the County name and its State
    {'$project': {'_id': 0, 'County':1, 'State': 1}},

    # Sort by State abbreviation
    {'$sort': {'State': 1}}
]))
```

	County	State
0	Lincoln County	AR
1	Lincoln County	CO
2	Lincoln County	GA
3	Lincoln County	ID
4	Lincoln County	KS
5	Lincoln County	KY
6	Lincoln Parish	LA
7	Lincoln County	ME
8	Lincoln County	MN
9	Lincoln County	MO
10	Lincoln County	MS
11	Lincoln County	MT
12	Lincoln County	NC
13	Lincoln County	NE
14	Lincoln County	NM
15	Lincoln County	NV
16	Lincoln County	OK
17	Lincoln County	OR
18	Lincoln County	SD
19	Lincoln County	TN
20	Lincoln County	WA
21	Lincoln County	WI
22	Lincoln County	WV
23	Lincoln County	WY

Once that I knew that the same County name was used many times, I still needed to check whether the same County name appeared more than once in the same State. Therefore, I identified a County not only by its County name but also by its State and its population number in 2014.

```
repeated_Counties = county.aggregate([
    # Group by County name, its State and its population in 2014
    # and count the number of time the same combination is repeated
    {"$group" : { "_id": {'county':"$County", 'state':'$State',
                          "population": "$Population.2014 Population"},
                 "count": { "$sum": 1 }}},

    # Keep only the ones which appear more than once
    {"$match": {"count" : {"$gt": 1}}},

    # Extract only the resulting number
    {"$project": {"count": "$count"}}
])

print("There are", len(pd.DataFrame(repeated_Counties)),
      "County in this collection which has the same name within the same State.")
```

```
There are 0 County in this collection which has the same name within the same State.
```

Since it was not the case, I was already sure that there were no duplicates in this dataset. A County could be uniquely identified using its name and its State. To be sure and demonstrate that assumption, I checked whether the same County name in a given State appeared more than once in the dataset.

```
duplicates = county.aggregate([
    # Group by County name, its State
    # and count the number of time the same combination is repeated
    {"$group" : { "_id": {'county':"$County", 'state':'$State'},
                 "count": { "$sum": 1 }}},

    # Keep only the ones which appear more than once
    {"$match": {"count" : {"$gt": 1}}},

    # Extract only the resulting number of duplicates
    {"$project": {"count": "$count"}}
])

print("There are", len(pd.DataFrame(duplicates)), "duplicate in this collection.")
```

```
There are 0 duplicate in this collection.
```

As expected, there were no duplicates in this census. A County could thus be uniquely identified by its County name and the State it belonged to.

4 Data analysis

For this project, I wanted to gain a better general understanding on different aspects of the population in the United States. Additionally, I focused on the geographical location of some of the ethnic groups in the United States, on the place of women in the society, on the population displacement as well as on the level of poverty in each County.

4.1 Count the number of Counties per State and the total population

In this section, I counted the number of Counties per State and summed up each State's total population for the years 2010 and 2014.

```
# Select the County name, State name, population of the County in 2010 and in 2014
project = {'$project': {'_id':0,
                        'County':1,
                        'State': 1,
                        'Population.2010 Population':1,
                        'Population.2014 Population':1}}

# Group by State, count the number of Counties
# and sum up the total population in 2010 and 2014 for each State
group = {'$group': {'_id':'$State',
                    'nb_counties': {'$sum': 1}, # number of Counties per State
                    'total_pop2010': {'$sum': '$Population.2010 Population'},
                    'total_pop2014': {'$sum': '$Population.2014 Population'}}}

# Sort the States by the total population in 2010, in descending order
sort = {'$sort': {'total_pop2010':-1}}

# Create the pipeline
pipeline = [project, group, sort]

# Perform the aggregation and create the cursor
cursor = county.aggregate(pipeline)

# Create the dataframe from the cursor
df = pd.DataFrame(cursor)

# Format the numbers for better visibility
df.loc[:, "total_pop2010"] = df["total_pop2010"].map('{:,d}'.format)
df.loc[:, "total_pop2014"] = df["total_pop2014"].map('{:,d}'.format)

# Set the State abbreviations as index and rename index column
df = df.set_index('_id').rename_axis('State')
df.rename(columns = {"_id":'State'}, inplace = True)

# For space reasons, I only print the first 10 States
df[:10]
```

	nb_counties	total_pop2010	total_pop2014
State			
CA	58	37,253,956	38,802,500
TX	254	25,145,561	26,956,958
NY	62	19,378,102	19,746,227
FL	67	18,801,310	19,893,297
IL	102	12,830,632	12,880,580
PA	67	12,702,379	12,787,209
OH	88	11,536,504	11,594,163
MI	83	9,883,640	9,909,877
GA	159	9,687,653	10,097,343
NC	100	9,535,483	9,943,964

To have more details about a given State and its Counties, I simply selected the State and displayed the results in a dataframe. For instance, I chose the State Delaware (DE). Additionally I projected the population percentage change between 2010 and 2014.

```
# Select the State
match = {'$match': {'State': 'DE'}}

# Select the columns that are of interest
project1 = {'$project':
            {'_id': 0,
             'County':1,
             'Population.Population Percent Change':1,
             'Population.2010 Population':1,
             'Population.2014 Population':1}}

# Group by County, get its population in 2010 and 2014
group = {'$group':
          {'_id':'$County',
           'Population_2010': {'$sum': '$Population.2010 Population'},
           'Population_2014': {'$sum': '$Population.2014 Population'},
           'Percent_change': {'$sum': '$Population.Population Percent Change'}}}

# Add "Percent_change" to compute the population growth in each County
project2 = {'$project': {"_id": 0,
                         "county": "$_id", # rename "_id" to "county"
                         'County':1,
                         'Population_2010':1,
                         'Population_2014':1,
                         'Percent_change': {'$divide': ['$Percent_change', 100]}}}

# Sort by County name
sort = {'$sort': {'county': 1}}

# Create the pipeline
pipeline = [match, project1, group, project2, sort]

# Create the cursor
cursor = county.aggregate(pipeline)

# Create the dataframe
df = pd.DataFrame(cursor)

# Format the numbers fo better readibility
df.loc[:, "Population_2010"] = df["Population_2010"].map('{:,d}'.format)
df.loc[:, "Population_2014"] = df["Population_2014"].map('{:,d}'.format)

# Rename the label axis
df = df.set_index('county').rename_axis('County')
df
```

	Population_2010	Population_2014	Percent_change
County			
Kent County	162,310	171,987	0.06
New Castle County	538,479	552,778	0.03
Sussex County	197,145	210,849	0.07

Between 2010 and 2014, the population increased for each County in Delaware.

4.2 Propotion of Ethnicities in each State

For this part of the analysis, I investigated the proportion of Asian, African-American (Black) and Hispanic/Latino people in each State. The objective was to know whether some ethnicities had State level preferences, or in other words, if they tended to live in specific States.

In order to visualise this, I wanted to obtain the proportion of each ethnicity per State.
I started by computing the approximate total population of the different groups for each County. To do so, I multiplied the County's percentage of each ethnic group by the County's total population for the year 2014. Nevertheless, in order to work with the right scale, I first had to divide the percentages by 100. Then, I computed the total population in each State, and the total number of people belonging to a given ethnic group for a given State. Once I had these numbers, I obtained the proportions of the different ethnicities for each State.

```
# Select the County, the State, the percentages of each Ethnicity
project1 = {
    "$project": { "_id": 0, "County":1, "State":1,
                 "Asian": "$Ethnicities.Asian Alone",
                 "Black":"$Ethnicities.Black Alone",
                 "Hispanic": "$Ethnicities.Hispanic or Latino",
                 "Pop2014": "$Population.2014 Population",

# Compute the approximate total number of Asian in each County
# First I rescale "Ethnicities.Asian Alone" by dividing by 100
# Then I multiply "Population.2014 Population" by the percentage
"total_asian": { "$multiply":
                [ { "$divide": # divide by 100 to have the right scale
                   [ "$Ethnicities.Asian Alone", 100 ] },
                 "$Population.2014 Population"]},

# Compute the approximate total number of African-American people in each County
"total_black": { "$multiply":
                [ { "$divide":
                   [ "$Ethnicities.Black Alone", 100 ] },
                 "$Population.2014 Population"]},

# Compute the approximate total number of Hispanic people in each County
"total_hispanic": { "$multiply":
                   [ { "$divide":
                      [ "$Ethnicities.Hispanic or Latino", 100 ] },
                    "$Population.2014 Population"]}}}

# Group by State and compute the total number of people per State,
# and per ethnicity in the given State
group = {"$group": {"_id": "$State", "totalPop": {"$sum": "$Pop2014"},
               "GTotal_asian": {"$sum": "$total_asian"},
               "GTotal_black": {"$sum": "$total_black"},
               "GTotal_hispanic": {"$sum": "$total_hispanic"}}}

# Compute the percentage of people of each ethnicity for a given State
project2 = {"$project": {"_id": 0,
                 "State": "$_id",
                 "prop_asian": { "$divide": ["$GTotal_asian", "$totalPop"] },
                 "prop_black": { "$divide": ["$GTotal_black", "$totalPop"] },
                 "prop_hispanic": { "$divide": ["$GTotal_hispanic", "$totalPop"] }}}


# Create the pipeline
pipeline = [project1, group, project2]

# Create the cursor
cursor = county.aggregate(pipeline)

# Create the dataframe
df = pd.DataFrame(cursor)
```

Below the resulting dataframe is presented. Amongst all States, I only kept and highlighted the States where the proportions were the smallest (lightpink) and the highest (green).

```
# Reformat the percentages for better readability
df["prop_asian"] = df["prop_asian"].map('{:.2f}'.format)
df["prop_black"] = df["prop_black"].map('{:.2f}'.format)
df["prop_hispanic"] = df["prop_hispanic"].map('{:.2f}'.format)

# Only keep and highlight the States where the proportions
# were the smallest (lightpink) and the highest (green)
df[(df["prop_asian"] == min(df["prop_asian"])) |
   (df["prop_asian"] == max(df["prop_asian"])) |
   (df["prop_black"] == min(df["prop_black"])) |
   (df["prop_black"] == max(df["prop_black"])) |
   (df["prop_hispanic"] == min(df["prop_hispanic"])) |
   (df["prop_hispanic"] == max(df["prop_hispanic"])
   )].style.highlight_min(subset=["prop_asian", # highlight the lowest proportions
                                  "prop_black",
                                  "prop_hispanic"],
                             color='lightpink'

   ).highlight_max(subset=["prop_asian", # highlight the highest proportions
                           "prop_black",
                           "prop_hispanic"],
                  color='lightgreen')
```

	State	prop_asian	prop_black	prop_hispanic
2	KY	0.01	0.08	0.03
5	ID	0.01	0.01	0.12
8	DC	0.04	0.49	0.10
9	NM	0.02	0.03	0.48
13	WV	0.01	0.04	0.01
24	WY	0.01	0.02	0.10
25	AR	0.01	0.16	0.07
28	ME	0.01	0.01	0.02
34	ND	0.01	0.02	0.03
39	NH	0.03	0.01	0.03
40	HI	0.37	0.02	0.10
41	AL	0.01	0.27	0.04
46	MS	0.01	0.38	0.03
47	SD	0.01	0.02	0.04
48	VT	0.02	0.01	0.02
49	MT	0.01	0.01	0.03
50	UT	0.02	0.01	0.14

It was interesting to underline the States with the highest proportions of these ethnicities:

- African American (Black) people made up 49% of the population in the District of Columbia (DC)
- 37% of Hawaiians (HI) were Asians
- Hispanic/Latino people made up 48% of New Mexico's (NM) population

On the other hand, Montana (MT), Maine (ME) and Idaho (ID) were amongst the States with the smallest ratio of both Asian and African American people. West Virginia (WV) was another example of a State with a low proportion of both Asian and Hispanic/Latino people.

4.3 Women in the United States

In this section, I focused on the place and the impact of women in the United States society. I extracted from this census the percentage of women in the general population and the percentage of companies owned by a woman in each County.

4.3.1 Correlation between the percentage of companies owned by a woman and the percentage of children

I investigated the correlation between having more women owning a company and the percentage of children in a given County. My first hypothesis was that if a woman owned a company, she would have had less time to have children, which would have impacted the percentage of children (*field name: Percent Under 18 Years*) in the County.

```
# Group by County
# Extract the percentage of women and the percentage of women owning a company
group = {"$group": {
    "_id": {"county": "$County", "state": "$State"},
    "Percent_women_owned": {"$sum": "$Employment.Firms.Women-Owned"},
    "Population_women": {"$sum": "$Miscellaneous.Percent Female"},
    "Percent_children": {'$sum': "$Age.Percent Under 18 Years"}}}

# Select only the fields of interest
project = {'$project': {"Percent_women_owned":1,
                        "Population_women":1,
                        "Percent_children":1}}

# Sort by the percentage of companies owned by a women,
# then by the percentage of women in the County, both in ascending order
sort = {'$sort': {'Percent_women_owned': 1, 'Population_women':1}}

# Have a column containing the County name
# and one column that contains the State abbreviation
project_group = {'$project':
                 {"_id": 0,
                  "county": "$_id.county", # rename to county
                  "state": "$_id.state", # rename to state
                  "Percent_women_owned":1,
                  "Population_women":1,
                  "Percent_children":1}}

# Creating the pipeline
pipeline = [group, project, project_group, sort]

# (Note: If we had: pipeline= [group, project, sort]
# we would have had a dictionary in the column _id,
# for example: {'county': 'Concho County', 'state': 'TX'}

# Create the cursor
cursor = county.aggregate(pipeline)

# Create the dataframe
df = pd.DataFrame(cursor)
df
```

	Percent_women_owned	Population_women	Percent_children	county	state
0	0.0	30.1	12.9	Concho County	TX
1	0.0	30.2	14.3	Crowley County	CO
2	0.0	30.9	10.1	Aleutians East Borough	AK
3	0.0	32.9	7.4	Forest County	PA
4	0.0	34.8	16.9	Wheeler County	GA
...	...	...	...	...	...
3138	48.3	53.3	25.3	Hopewell city	VA
3139	49.3	49.2	23.3	Los Alamos County	NM
3140	52.9	50.9	23.3	Rooks County	KS
3141	53.1	50.2	23.2	Jefferson County	MS
3142	56.2	48.8	24.4	Camas County	ID

3143 rows × 5 columns

It was interesting to note that Concho County in Texas (TX) and Crowley County in Colorado (CO) had no companies owned by women.

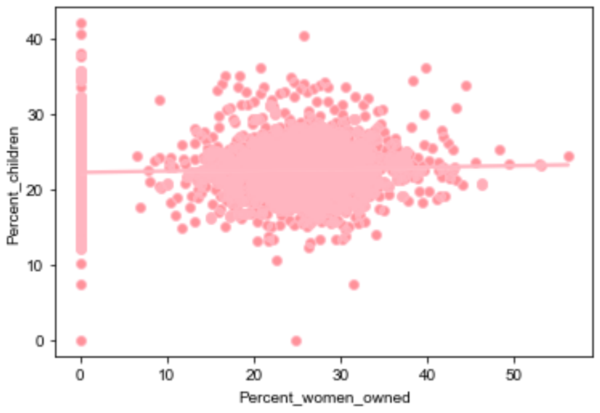
Based on the dataframe created above, I visualised the correlation between the percentage of women owning a company and the percentage of people being less than 18 years old in the County.

```
# Draw a scatter plot
df.plot.scatter(x ="Percent_women_owned",
                y ="Percent_children",
                c ="red")

# Remove the background
sns.set(style="white", color_codes=True)

# Add a regression line
sns.regplot(x='Percent_women_owned',y='Percent_children',
            data=df, fit_reg=True, color="lightpink")
```

```
<AxesSubplot:xlabel='Percent_women_owned', ylabel='Percent_children'>
```



As seen above, it seemed that there were no correlation between the percentage of women owning a company and the percentage of children in the County. Indeed, the slope was almost flat.

4.3.2 Percentage of women owning a company: Comparison between Counties

Based on the dataframe created above, I was surprised to see that some Counties had no companies owned by women. Therefore, I wanted to know which State had the highest number of Counties having no women owning a company. In a second step, I also seeked to find out which State had the highest percentage of companies owned women.

To have an idea of the percentage range, I first searched for the extreme values, namely the minimum and the maximum percentage of companies owned by a woman, across all Counties in the United States.

```
cursor = county.aggregate([
    {"$project": {"_id":0, "County":1, "State":1, "Women_Owned": "$Employment.Firms.Women-Owned"}},

    {"$group": {"_id": {},
                "min": {"$min": "$Women_Owned"}, # select the lowest percentage
                "max": {"$max": "$Women_Owned"}}}, # select the highest percentage

    {'$project':
                  {"_id": 0}}
])
pd.DataFrame(cursor)
```

	min	max
0	0.0	56.2

There were at least one County in which 56.2% of the companies were owned by women. Following this finding, I computed the number of Counties that had no companies owned by women, or in other words, Counties whose companies were only owned by men. Finally, I repeated the logic for the Counties whose at least 50% of their companies were owned by a woman.

```
# Select the County, the State and the percentage of companies owned by a woman
project = {"$project": {"_id":0, "County":1, "State":1, "Women_Owned": "$Employment.Firms.Women-Owned"}}

# Create new fields
new_fields = {"$addFields": {
    # Women_Owned_eq0: is equal to 1
    # when the County has no companies owned by a woman, 0 otherwise
    "Women_Owned_eq0": {"$cond": {"if": {"$eq": ["$Women_Owned", 0]}, "then": 1, "else": 0}},
    # Women_Owned_gte50: is equal to 1 when the County
    # has at least 50% of its companies owned by a woman
    "Women_Owned_gte50": {"$cond": {"if": {"$gte": ["$Women_Owned", 50]}, "then": 1, "else": 0}}}}

# Group by State, sum up the number of Counties having 0% of their companies owned by a woman
# and also sum up the number of Counties having 50% or more of their companies owned by a woman
group = {"$group": {"_id": "$State",
               "nb_counties": {"$sum": 1},
               "equal_0": {"$sum": "$Women_Owned_eq0"},
               "more50": {"$sum": "$Women_Owned_gte50"}}}

# Sort by the number of Counties having no companies owned by a woman
sort = {"$sort": {"equal_0":-1}}

# Create the pipeline
pipeline = [project, new_fields, group, sort]

# Create the cursor
cursor = county.aggregate(pipeline)

df = pd.DataFrame(cursor)
```

For better readability, I only printed the top five States having the largest number of Counties that had no companies owned by a woman.

```
# Limit to 5 results
limit = {"$limit": 5}

# Create a new pipeline, with the limit
pipeline_limit = [project, new_fields, group, sort, limit]

# Create the cursor
cursor = county.aggregate(pipeline_limit)

# Create the dataframe
df2 = pd.DataFrame(cursor)
df2
```

	_id	nb_counties	equal_0	more50
0	TX	254	131	0
1	NE	93	58	0
2	KS	105	51	1
3	KY	120	43	0
4	GA	159	42	0

Texas (TX) and Kansas (KS) had about half of their Counties whose companies were only men-owned. For Nebraska (NE) it goes up to more than two thirds.

Remark: Such results need to be interpreted with caution. It is possible that the numbers were not recorded correctly. Indeed, there might be a chance that if such information was not available, the percentage was automatically set to zero.

Finally, I compared these findings with the number of Counties having at least 50% of their companies owned by women.

```
# Count the number of States having at least one County,
# which had at least 50% of its companies owned by women
numbr = len(df[df["more50"] > 0].index)
numbr

# This number was then used to limit the results in the pipeline
```

```
3
```

```
# Sort by the number of Counties, in descending order
sort2 = {"$sort": {"more50":-1, "equal_0":-1}}

# Limit the results to the number found above
limit2 = {"$limit": numbr}

# Create the pipeline
pipeline_limit = [project, new_fields, group, sort2, limit2]

# Create the cursor
cursor = county.aggregate(pipeline_limit)

# Create the dataframe
df3 = pd.DataFrame(cursor)

# Rename _id to State
df3.rename(columns = {"_id":'State'}, inplace = True)

df3
```

	State	nb_counties	equal_0	more50
0	KS	105	51	1
1	MS	82	16	1
2	ID	44	14	1

There were only 3 Counties in the United States, one in Kansas (KS), one in Mississippi (MS) and one in Idaho (ID) that had at least 50% of their companies owned by women.

4.4 Displacement of the County's population

In this section of the analysis, I was interested in discovering which County in each State had the highest and the lowest percentage of displacement amongst its population. To answer this question I used the field *Living in Same House +1 Years*.

Note: the United States Census Bureau defined the field as follows:
Living in Same House +1 Years: the percentage of people living in the same house last year

```
# Group by County and select the percentage
group1 = {"$group": {"_id": {"State": "$State",
                             "County": "$County"},
                     "Living_gte_1y": {"$sum":"$Miscellaneous.Living in Same House +1 Years"}}}

# Sort by percentage in ascending order
sort1 = {"$sort": {"Living_gte_1y":1}}

# By State, select the Counties with the highest and the lowest percentage
group2 = {"$group": {"_id": "$_id.State",
                     "Longest_time_county":  {"$last": "$_id.County"},
                     "Longest_percent":   {"$last": "$Living_gte_1y"},
                     "Shortest_time_county": {"$first": "$_id.County"},
                     "Shortest_percent":  {"$first": "$Living_gte_1y"}}}

# Select the States, their "top"/"worst" Counties and their percentage
project = {"$project": {"_id": 0,
                  "State": "$_id",
                  "Longest_time_county":1,
                  "Longest_percent":1,
                  "Shortest_time_county":1,
                  "Shortest_percent": 1}}

# Sort by the Counties having the highest percentage
sort2 = {"$sort": {"Longest_percent": -1}}

# Create the pipeline
pipeline = [group1, sort1, group2, project, sort2]

# Create the cursor
cursor = county.aggregate(pipeline)

# Create the dataframe
df = pd.DataFrame(cursor)
df

# Highlight the County, with lowest percent (lightpink)
df.style.highlight_min(subset='Shortest_percent', color='lightpink')
```

	Longest_time_county	Longest_percent	Shortest_time_county	Shortest_percent	State
0	Kenedy County	99.80	Brazos County	68.00	TX
1	Conecuh County	96.90	Lee County	76.40	AL
2	Wheeler County	96.40	Chattahoochee County	50.80	GA
3	Sioux County	96.40	Dawes County	75.80	NE
4	McCone County	96.20	Gallatin County	75.20	MT
5	Slope County	95.80	Ward County	77.20	ND
6	Northumberland County	95.70	Williamsburg city	59.10	VA
7	Grundy County	95.30	Montgomery County	76.30	TN
8	Ontonagon County	95.00	Isabella County	68.20	MI
9	Cameron Parish	95.00	Lincoln Parish	77.20	LA
10	Cleveland County	94.70	Washington County	75.20	AR
11	Alleghany County	94.60	Watauga County	70.20	NC
12	Holmes County	94.50	Athens County	70.30	OH
13	Wirt County	94.30	Gilmer County	74.50	WV
14	Amite County	94.30	Oktibbeha County	71.40	MS
15	Jackson County	94.00	Clay County	65.00	SD
16	Richmond County	93.90	Tompkins County	73.90	NY
17	Carlisle County	93.80	Fayette County	75.50	KY
18	Rawlins County	93.80	Douglas County	71.60	KS
19	Sierra County	93.60	Lassen County	71.10	CA
20	Socorro County	93.50	Union County	75.70	NM

	Longest_time_county	Longest_percent	Shortest_time_county	Shortest_percent	State
21	Kewaunee County	93.50	Dunn County	78.10	WI
22	Carroll County	93.40	Wicomico County	81.00	MD
23	Norman County	93.40	Stevens County	79.00	MN
24	Calhoun County	93.30	Richland County	77.90	SC
25	Roger Mills County	93.30	Payne County	70.90	OK
26	Elk County	93.20	Centre County	72.40	PA
27	Piute County	93.10	Utah County	78.10	UT
28	Osage County	92.90	Pulaski County	59.30	MO
29	Sussex County	92.70	Hudson County	85.90	NJ
30	Benton County	92.60	Story County	67.80	IA
31	Clay County	92.60	McDonough County	70.60	IL
32	Dukes County	92.60	Suffolk County	79.00	MA
33	Mineral County	92.30	Gunnison County	73.30	CO
34	Pend Oreille County	92.10	Whitman County	65.40	WA
35	Oneida County	92.00	Madison County	60.70	ID
36	Orange County	91.90	Chittenden County	81.50	VT
37	Apache County	91.70	Mohave County	79.00	AZ
38	Piscataquis County	91.30	Penobscot County	84.00	ME
39	Litchfield County	91.30	New London County	85.20	CT
40	LaGrange County	91.30	Monroe County	67.70	IN
41	Hood River County	90.70	Lake County	77.80	OR
42	Kent County	90.10	Newport County	83.80	RI
43	Kalawao County	90.10	Maui County	82.80	HI
44	Jefferson County	90.10	Leon County	72.80	FL
45	Johnson County	90.00	Albany County	68.80	WY
46	Storey County	89.70	Clark County	76.80	NV
47	Rockingham County	89.60	Strafford County	80.50	NH
48	Wade Hampton Census Area	89.50	Fairbanks North Star Borough	73.80	AK
49	Sussex County	88.00	New Castle County	85.60	DE
50	District of Columbia	80.60	District of Columbia	80.60	DC

Almost all the population in Kenedy County, in Texas (TX) lived in the same place since the previous year.

In contrast, about half of the population in Chattahoochee County in Georgia (GA) moved out the year before.

Note: The District of Columbia, in DC, had the same percentage for both columns, since it had only one County.

4.5 States with high poverty

Finally, I wanted to learn more about the geographical distribution of poverty in the United States. To answer such question, the census provided the percentage of the County's population living below the poverty level. The higher the percentage was, the worse the population's financial status was.

I sorted the Counties into buckets according to the County's percentage of people living below the poverty level. I grouped the results by State and counted the number of Counties in each bucket. Finally, I ordered the results by State according to the number of high-poverty Counties.

As a note, in order to choose the right bucket sizes for appropriate classification, I had to know the minimum and maximum values. They are displayed below.

```
c = county.aggregate([
    {"$project": {"_id":0, "County":1, "State":1,
                  "Below_Poverty": "$Income.Persons Below Poverty Level"}},

    {"$group": {"_id": {},
               # minimum percentage of people below poverty level
               "min": {"$min": "$Below_Poverty"},
               # maximum percentage of people below poverty level
               "max": {"$max": "$Below_Poverty"}}},
    {'$project':
                 {"_id": 0}}
])

pd.DataFrame(c)
```

	min	max
0	0.9	53.2

```python
# Select the County, the State, the percentage of people below poverty level
project = {"$project": {"_id":0, "County":1, "State":1,
                        "Below_Poverty": "$Income.Persons Below Poverty Level"}}

# Create new fields for the classification buckets
new_fields = {"$addFields": {
    # Is equal to 1 if the County has at most 5% of its population below poverty level
    "Below_Poverty_lte5": {"$cond": {"if":
                                 {"$lte": ["$Below_Poverty", 5]},
                                 "then": 1, "else": 0}},

    # Is equal to 1 if the County has less than 10%
    # but more than 5% of its population below poverty level
    "Below_Poverty_gt5_lt10": {"$cond": {"if":
                                     {"$and": [{"$gt": ["$Below_Poverty", 5]},
                                               {"$lt": ["$Below_Poverty", 10]}]},
                                     "then": 1, "else": 0}},
    # Between 10% and 20%
    "Below_Poverty_gte10_lt20": {"$cond": {"if":
                                       {"$and": [{"$gte": ["$Below_Poverty", 10]},
                                                 {"$lt": ["$Below_Poverty", 20]}]},
                                       "then": 1, "else": 0}},
    # Between 20% and 30%
    "Below_Poverty_gte20_lt30": {"$cond": {"if":
                                       {"$and": [{"$gte": ["$Below_Poverty", 20]},
                                                 {"$lt": ["$Below_Poverty", 30]}]},
                                       "then": 1, "else": 0}},
    # Between 30% and 40%
    "Below_Poverty_gte30_lt40": {"$cond": {"if":
                                       {"$and": [{"$gte": ["$Below_Poverty", 30]},
                                                 {"$lt": ["$Below_Poverty", 40]}]},
                                       "then": 1, "else": 0}},
    # Between 40% and 50%
    "Below_Poverty_gte40_lt50": {"$cond": {"if":
                                       {"$and": [{"$gte": ["$Below_Poverty", 40]},
                                                 {"$lt": ["$Below_Poverty", 50]}]},
                                       "then": 1, "else": 0}},
    # At least 50% of the County's population lives below the poverty level
    "Below_Poverty_gte50": {"$cond": {"if":
                                  {"$gte": ["$Below_Poverty", 50]},
                                  "then": 1, "else": 0}}}}

# Group by State, counting the number of Counties in each bucket
group = {"$group": {"_id": "$State",
                    "nb_counties": {"$sum": 1},
                    "gte50": {"$sum": "$Below_Poverty_gte50"},
                    "gte40_lt50": {"$sum": "$Below_Poverty_gte40_lt50"},
                    "gte30_lt40": {"$sum": "$Below_Poverty_gte30_lt40"},
                    "gte20_lt30": {"$sum": "$Below_Poverty_gte20_lt30"},
                    "gte10_lt20": {"$sum": "$Below_Poverty_gte10_lt20"},
                    "gt5_lt10": {"$sum": "$Below_Poverty_gt5_lt10"},
                    "lte_5": {"$sum": "$Below_Poverty_lte5"}}}

# Sort by the number of Counties in a State having more than 50%
# of its population below poverty level
# then between 40% and 50% and then between 30% and 40%, all in descending order
sort = {"$sort": {"gte50":-1, "gte40_lt50":-1, "gte30_lt40":-1}}

# Limit the number of results for space reasons
limit = {"$limit": 10}

# Create the pipeline
pipeline = [project, new_fields, group, sort, limit]

# Create the cursor
cursor = county.aggregate(pipeline)




# Create the dataframe
df = pd.DataFrame(cursor)
df
```

	_id	nb_counties	gte50	gte40_lt50	gte30_lt40	gte20_lt30	gte10_lt20	gt5_lt10	lte_5
0	SD	66	1	3	4	6	40	9	3
1	MS	82	0	5	14	46	16	1	0
2	TX	254	0	2	9	64	157	19	3
3	LA	64	0	2	3	26	32	1	0
4	GA	159	0	1	22	78	52	6	0
5	KY	120	0	1	12	51	53	3	0
6	ND	53	0	1	2	0	25	24	1
7	AL	67	0	0	5	30	31	1	0
8	AR	75	0	0	5	43	25	2	0
9	SC	46	0	0	4	21	21	0	0

South Dakota (SD) had one County where at least 50% of its population was living below the poverty threshold. And 3 Counties where the proportion was between 40% and 50%. In order to better visualise the extreme values, I selected the Counties with the highest and the lowest percentages of people living below the poverty level.

```
cursor = county.aggregate([
    # Select the Counties having more than 50% of its population below poverty level
    {"$match": {"$or": [{"Income.Persons Below Poverty Level": {"$gte": 50}},
                        # or less than 5% of its population below poverty level
                        {"Income.Persons Below Poverty Level": {"$lte": 5}}]}},

    # Select the Counties and their percentage
    {"$project": {"_id": 0, "County":1, "State":1,
                  "Poverty_Level":"$Income.Persons Below Poverty Level"}},

    # Order by the percentage of poverty level, in descending order
    {"$sort": {"Poverty_Level":-1}}
])

pd.DataFrame(cursor)
```

	County	State	Poverty_Level
0	Shannon County	SD	53.2
1	Hendricks County	IN	5.0
2	Carver County	MN	5.0
3	Somerset County	NJ	5.0
4	Calvert County	MD	4.9
5	Kearney County	NE	4.9
6	Cavalier County	ND	4.9
7	Delaware County	OH	4.9
8	Kendall County	IL	4.8
9	Deuel County	SD	4.7
10	Howard County	MD	4.6
11	Lincoln County	SD	4.5
12	Morris County	NJ	4.4
13	Los Alamos County	NM	4.4
14	Union County	SD	4.4
15	Glasscock County	TX	4.2
16	Stanton County	KS	4.0
17	Hunterdon County	NJ	4.0
18	Morgan County	UT	4.0
19	Falls Church city	VA	4.0
20	Douglas County	CO	3.9
21	Loudoun County	VA	3.6
22	Roberts County	TX	3.0
23	Borden County	TX	0.9

South Dakota (SD) had Counties with high and low percentages of population living below poverty level.

5 Conclusions

In conclusion, working with census data was extremely interesting and helped me to get some insights about the makeup of the population of the United States.

First of all, I discovered that County names were not unique, for instance there were 30 Counties by the name of "Washington County". Additionally, I found out that share of different ethnic groups varies greatly by State. I also analysed the impact of women in American society and discovered that some Counties have no women-owned companies. This insight is extremely valuable and should be acted upon. The United States should ensure equal chances to people of different genders to become business owners.

I learned that the rate at which inhabitants change homes differs widely. However, the data available is probably too general. This because it does not provide information on whether a household moves to another County or only changes to another location within the same County. This information could be interesting for County governments, which need to understand migration patterns in their region.

Finally, in the last part of my analysis, I divided the Counties by the proportion of people living below the poverty threshold. I also selected the Counties with the highest and lowest levels of poverty across the entire United States. Such information is essential to understand the geographical distribution of poverty in the United States. Action needs to be taken in order to remedy these regional inequalities by investing in the communities that are in the greatest need.

These findings have given me the inspiration for further projects. Examples could be working with more granular data at the city or neighbourhood level. This would allow me to draw more precise conclusions on the American society.

6 Learnings

While attending this course and working on this project, I was able to broaden my knowledge of databases, data modelling and of Python. First of all, I learned how to perform SQL queries using Python. It is extremely useful as it enables me to use a single working environment for fetching, analysing and modelling data.

Additionally, I gained the ability to work with unstructured datasets and non-relational databases, which are important skills given their widespread use in companies.

I also discovered a new tool, MongoDB, a NoSQL database which allows me to easily work with JSON-like files. Thus, by using PyMongo tool, I am now able to understand, analyse and filter data on MongoDB efficiently and directly from Python. PyMongo enables me to retrieve documents from MongoDB based on various filters, to create new variables, to aggregate fields using a pipeline, and finally to visualise the result in a Pandas dataframe or through a plot.

In this project, I really valued being able to apply my new knowledge on a real case. The biggest challenges were also the most interesting ones. They taught me how to deal with nested fields and how to create new fields based on conditions.

All in all, I was very satisfied with how much I was able to learn, the challenges that I overcame and the insights I gained through this project. Looking forward, I am excited to continue to develop my competences related to managing SQL and NoSQL databases even further with other projects.