# DataEng S24: Data Integration In-class Assignment

This week you will gain hands-on experience with Data Integration by combining data from two distinct sources into a unified DataFrame for analysis.

**Submit**: Make a copy of this document and use it to record your responses and results. Use colored highlighting for your responses and results. Store a PDF copy of the document in your git repository along with any needed code before submitting for this week.

Your job is to integrate **county-level COVID-19 data** with the **ACS Census Tract data for 2017** to build a model that allows you to relate COVID numbers with economic data such as population, per capita income and poverty level. To do this you should build two pandas DataFrames as follows.

County_Info: demographic summaries for each county. This table should have one row per county (there are more than 3000 counties in the USA), and each row must include the following columns:

Name - name of the county
State - name of the state in which the county resides
Population - population of this county
Poverty - % of people in poverty in this county
PerCapitaIncome - per capita personal income for this county
ID - a unique integer ID for the county. You may choose this ID any way you like, it just needs to be unique among all counties and needs to be referenced by the COVID_monthly rows (foreign key lookup).

COVID_monthly: data about COVID cases and deaths for each USA county.

ID - refers to the county found in the County_info table (i.e., this is a foreign key lookup)
Month - integer in range 1..12 indicating the month of the year
Cases - number of COVID cases recorded for the corresponding county during the corresponding month
Deaths - number of COVID-related deaths recorded for the corresponding county during the corresponding month

For this activity you should use whichever development environment is convenient for you to develop with python and pandas. Google Colab and Jupyter are good choices. You are not required to use GCP, but you can use it if you prefer.

Submit: by Friday at 10pm

## A. [MUST] Discussion Question

Within your group, identify two or more sources of data that might be integrated to analyze a problem that could not be easily analyzed or solved otherwise. The data sources and problems do not need to be related to anything we have done in class and do not even need to be serious. Be imaginative; you do not need to describe how to integrate the data sets.

For example, you might say something like, “Integrate historic weather/precipitation data with crop yield data to help develop a system for anticipating future prices of corn and wheat.”

Sports analytics is something I am personally very interested in, specifically with basketball and football. When looking at predicting success of rookies coming into the league there are so many aspects that we like to look at- which requires data integration. Most commonly we like to look at past players. We can integrate their college statistics with their professional statistics, physical characteristics, and combine results. Taking in and analyzing these datasets holistically we can create models to better understand what features more likely indicate success for new rookies.

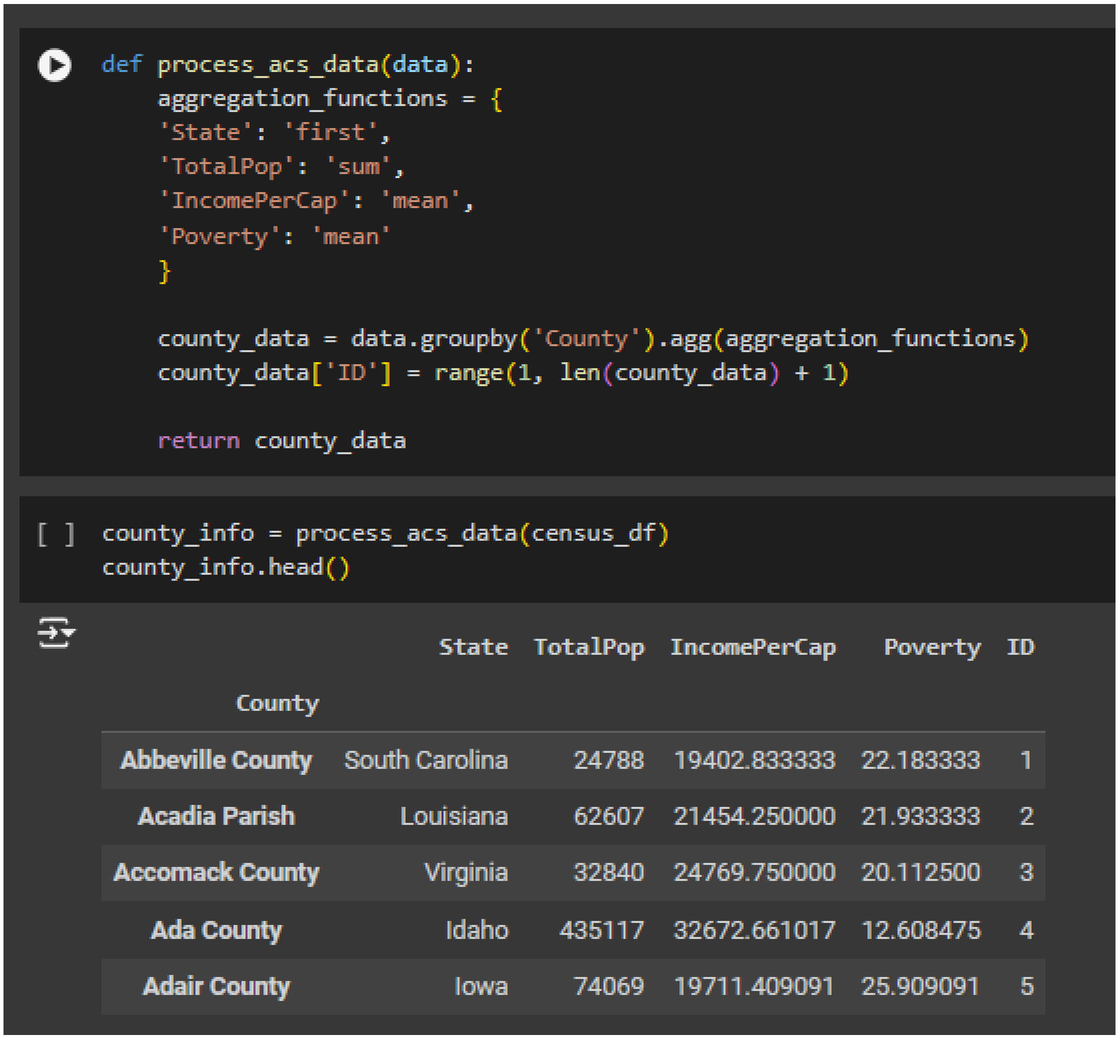
## B. [MUST] Transform the ACS Data

The ACS data is separated into “Census Tracts” which are regions within counties that correspond to groups of approximately 4000 people. The Census Bureau defines these to help organize the actual job of collecting census data. I.e., they did it this way for their convenience, not yours, and this grouping makes your Data Engineering job more challenging. Your “dimension of overlap” is the county not the census tract, so you must transform the ACS data.

To properly integrate the ACS with the COVID data, aggregate the per-tract data to the county level of resolution.

Create a python+pandas program that transforms the ACS data into a one-row-per-county ACS DataFrame called County_info. To do this you will need to think about how to properly aggregate Census Tract-level data into County-level summaries. Your transformation code should also eliminate unneeded columns from the ACS data.

Also, add an ID column to your County_info dataframe so that the county can be referred to by the COVID data.



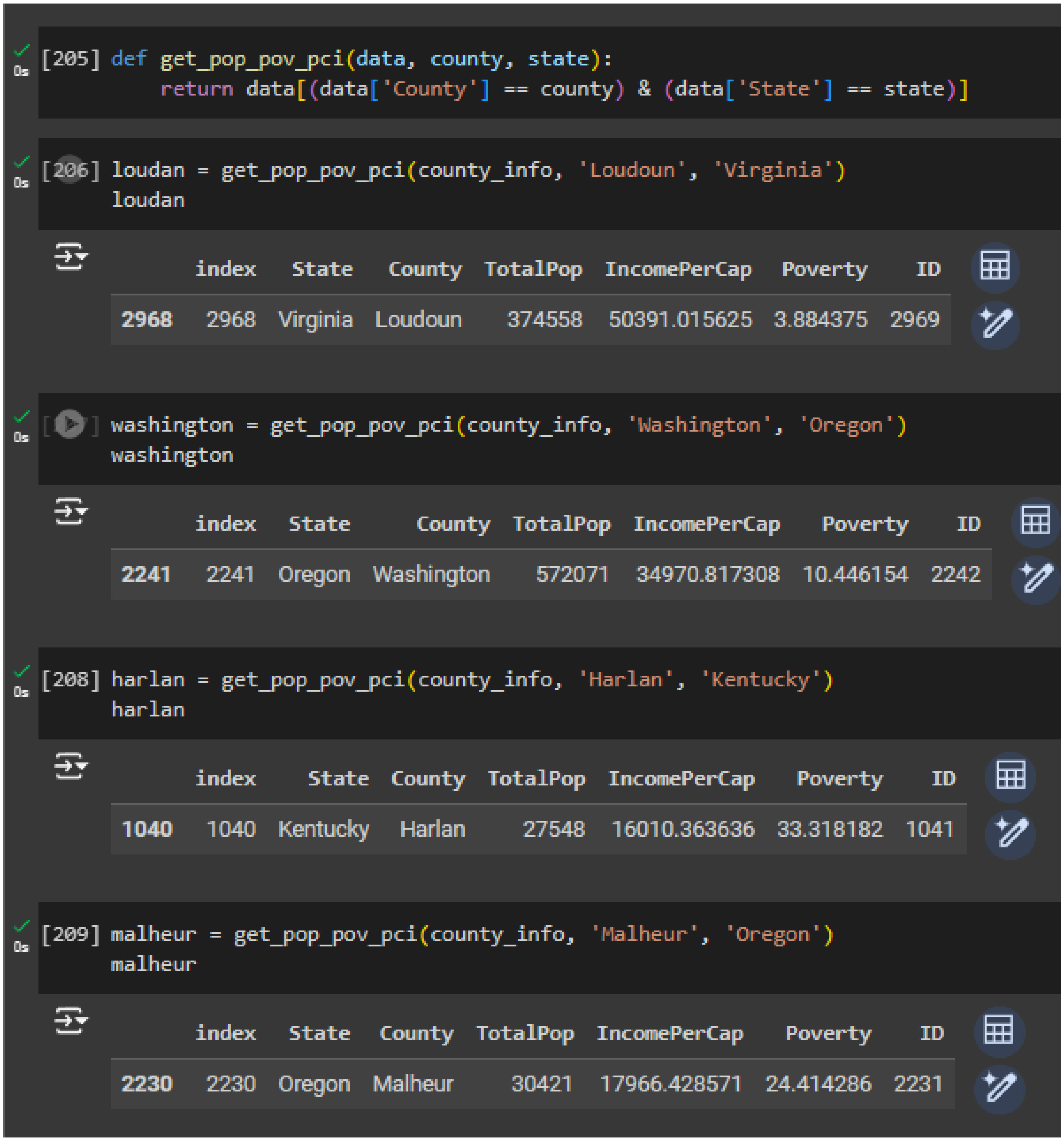
```
def process_acs_data(data):
    aggregation_functions = {
    'State': 'first',
    'TotalPop': 'sum',
    'IncomePerCap': 'mean',
    'Poverty': 'mean'
    }

    county_data = data.groupby('County').agg(aggregation_functions)
    county_data['ID'] = range(1, len(county_data) + 1)

    return county_data
```

```
county_info = process_acs_data(census_df)
county_info.head()
```

```
                         State  TotalPop  IncomePerCap    Poverty  ID
           County
 Abbeville County  South Carolina    24788  19402.833333  22.183333   1
    Acadia Parish       Louisiana    62607  21454.250000  21.933333   2
 Accomack County         Virginia    32840  24769.750000  20.112500   3
       Ada County           Idaho   435117  32672.661017  12.608475   4
     Adair County            Iowa    74069  19711.409091  25.909091   5
```

Fill the following table using data from your County_info DataFrame:

| County | Population | %poverty | PerCapitaIncome |
|---|---|---|---|
| Loudoun County, Virginia | 374558 | 3.88 | 50391.01 |
| Washington County, Oregon | 572071 | 10.44 | 34970.81 |
| Harlan County, Kentucky | 27548 | 33.31 | 16010.36 |
| Malheur County, Oregon | 30421 | 24.41 | 17966.42 |



```
[205] def get_pop_pov_pci(data, county, state):
          return data[(data['County'] == county) & (data['State'] == state)]
```

```
[206] loudan = get_pop_pov_pci(county_info, 'Loudoun', 'Virginia')
      loudan
```

```
        index  State     County   TotalPop  IncomePerCap  Poverty   ID
2968    2968   Virginia  Loudoun    374558  50391.015625  3.884375  2969
```

```
[▶7] washington = get_pop_pov_pci(county_info, 'Washington', 'Oregon')
     washington
```

```
        index  State     County      TotalPop  IncomePerCap  Poverty    ID
2241    2241   Oregon    Washington    572071  34970.817308  10.446154  2242
```

```
[208] harlan = get_pop_pov_pci(county_info, 'Harlan', 'Kentucky')
      harlan
```

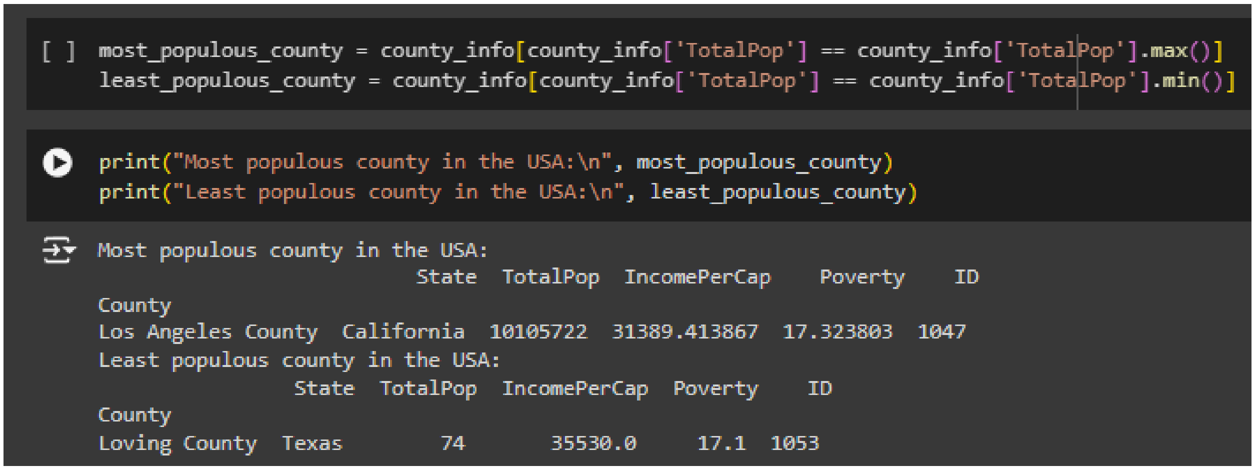
```
        index  State     County   TotalPop  IncomePerCap  Poverty    ID
1040    1040   Kentucky  Harlan      27548  16010.363636  33.318182  1041
```

```
[209] malheur = get_pop_pov_pci(county_info, 'Malheur', 'Oregon')
      malheur
```

```
        index  State     County   TotalPop  IncomePerCap  Poverty    ID
2230    2230   Oregon    Malheur     30421  17966.428571  24.414286  2231
```

Answer the following questions:

- Most populous county in the USA?
- Least populous county in the USA?



```
[ ]  most_populous_county = county_info[county_info['TotalPop'] == county_info['TotalPop'].max()]
     least_populous_county = county_info[county_info['TotalPop'] == county_info['TotalPop'].min()]
```

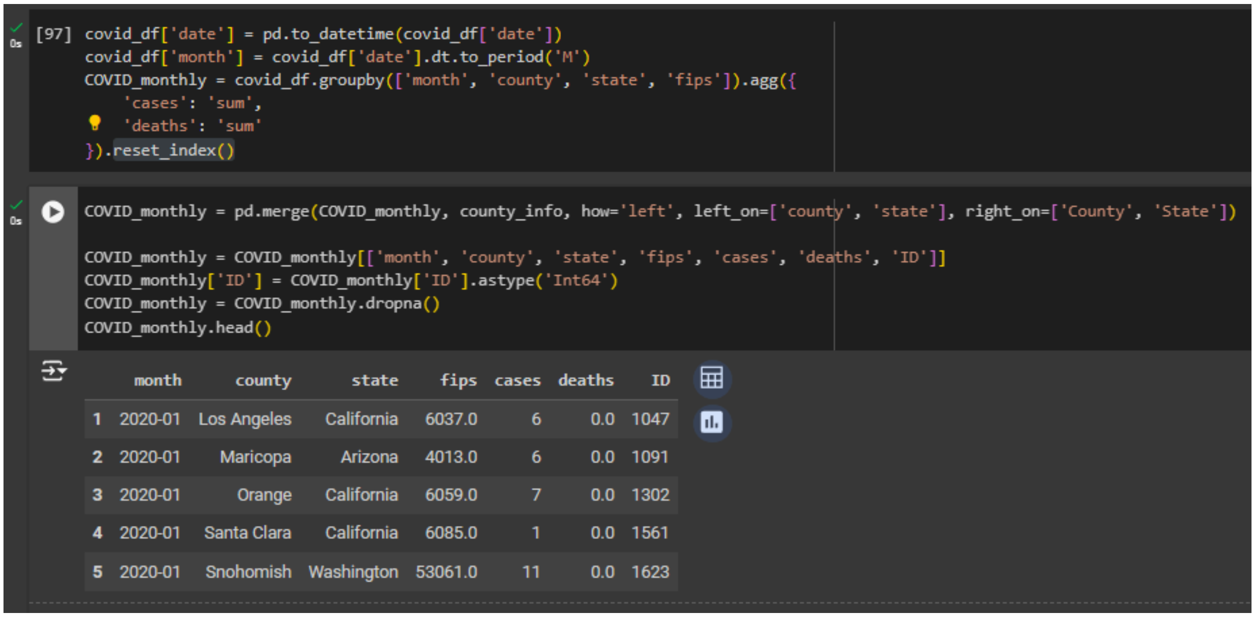
```
print("Most populous county in the USA:\n", most_populous_county)
print("Least populous county in the USA:\n", least_populous_county)
```

```
Most populous county in the USA:
                          State  TotalPop  IncomePerCap    Poverty    ID
County
Los Angeles County  California  10105722  31389.413867  17.323803  1047
Least populous county in the USA:
                 State  TotalPop  IncomePerCap  Poverty    ID
County
Loving County  Texas        74       35530.0     17.1  1053
```

## C. [MUST] Transform the COVID Data

Simplify the COVID data along the time dimension. The COVID data set contains day-level resolution data from (approximately) January of 2020 through February of 2021. From this you should produce a **COVID_monthly** Data Frame that has just one row per month per county.

Also, add a county ID column that is a foreign key lookup to the corresponding ID column in the County_info DataFrame.



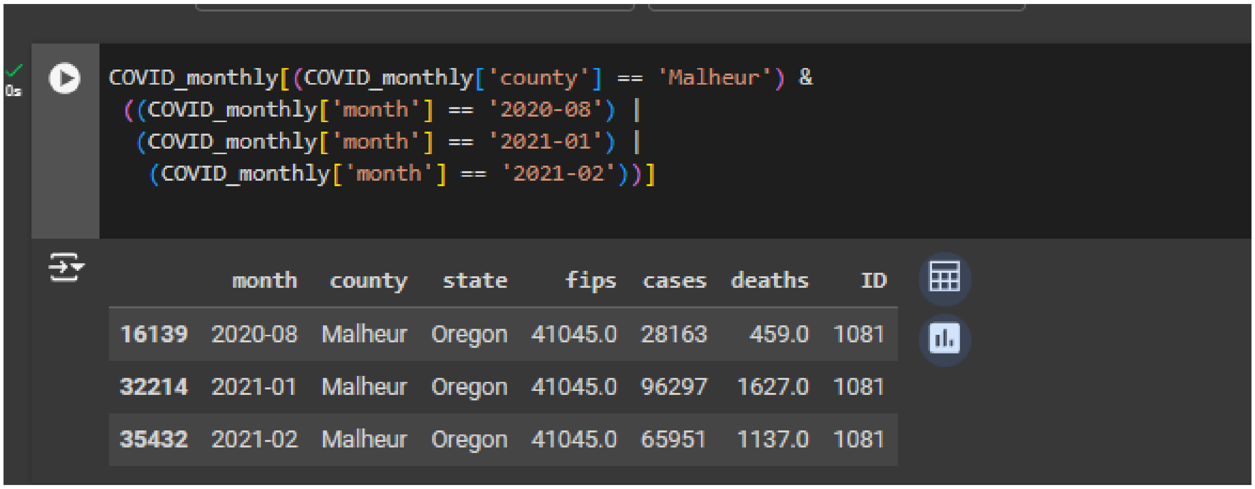
```
[97] covid_df['date'] = pd.to_datetime(covid_df['date'])
     covid_df['month'] = covid_df['date'].dt.to_period('M')
     COVID_monthly = covid_df.groupby(['month', 'county', 'state', 'fips']).agg({
         'cases': 'sum',
         'deaths': 'sum'
     }).reset_index()
```

```
COVID_monthly = pd.merge(COVID_monthly, county_info, how='left', left_on=['county', 'state'], right_on=['County', 'State'])

COVID_monthly = COVID_monthly[['month', 'county', 'state', 'fips', 'cases', 'deaths', 'ID']]
COVID_monthly['ID'] = COVID_monthly['ID'].astype('Int64')
COVID_monthly = COVID_monthly.dropna()
COVID_monthly.head()
```

|   | month | county | state | fips | cases | deaths | ID |
|---|---|---|---|---|---|---|---|
| 1 | 2020-01 | Los Angeles | California | 6037.0 | 6 | 0.0 | 1047 |
| 2 | 2020-01 | Maricopa | Arizona | 4013.0 | 6 | 0.0 | 1091 |
| 3 | 2020-01 | Orange | California | 6059.0 | 7 | 0.0 | 1302 |
| 4 | 2020-01 | Santa Clara | California | 6085.0 | 1 | 0.0 | 1561 |
| 5 | 2020-01 | Snohomish | Washington | 53061.0 | 11 | 0.0 | 1623 |

Fill the following table using data from your COVID_monthly DataFrame:

| County | Month | # cases | # deaths |
|---|---|---|---|
| Malheur County, Oregon | August 2020 | 28163 | 459.0 |
| Malheur County, Oregon | January 2021 | 96297 | 1627.0 |
| Malheur County, Oregon | February 2021 | 65951 | 1137.0 |

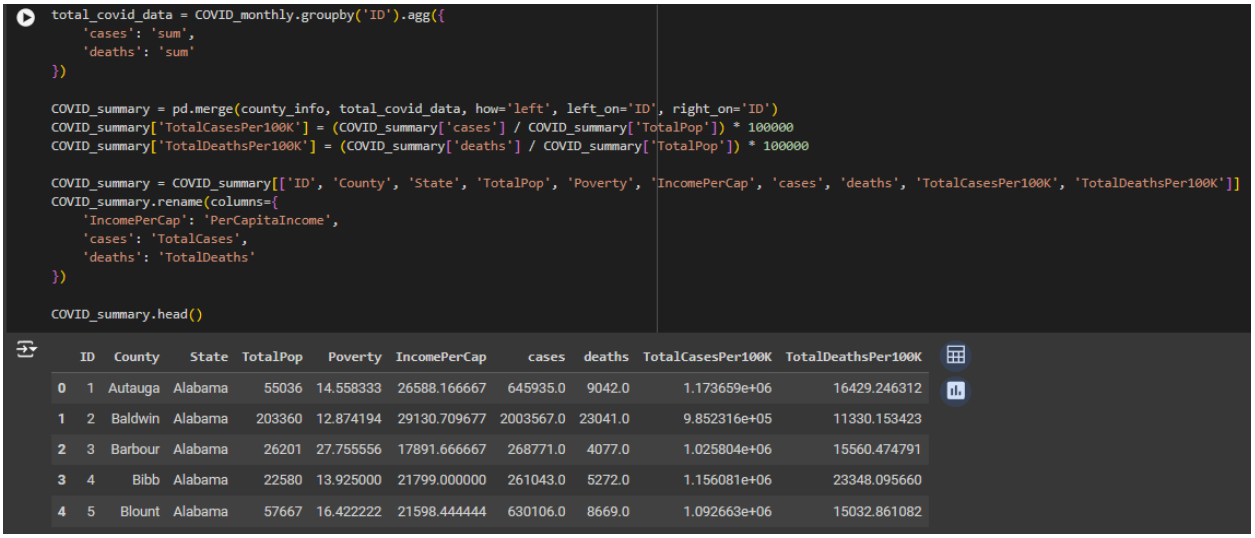


```
COVID_monthly[(COVID_monthly['county'] == 'Malheur') &
 ((COVID_monthly['month'] == '2020-08') |
  (COVID_monthly['month'] == '2021-01') |
   (COVID_monthly['month'] == '2021-02'))]
```

```
         month   county   state     fips   cases  deaths    ID
16139  2020-08  Malheur  Oregon  41045.0  28163   459.0  1081
32214  2021-01  Malheur  Oregon  41045.0  96297  1627.0  1081
35432  2021-02  Malheur  Oregon  41045.0  65951  1137.0  1081
```

## D. [MUST] Integrate COVID Data with ACS Data

Create a new single pandas DataFrame called COVID_summary containing one row per county. It should have these columns:
ID - integer identifier for the county
Population, Poverty, PerCapitaIncome - these should all be the same as from the County_info DataFrame
TotalCases, TotalDeaths - these two values should come from the COVID_monthly data and summed over all months
TotalCasesPer100K, TotalDeathsPer100K - these two values should be computed by dividing the the TotalCases and TotalDeaths (respectively) by (Population / 100000)

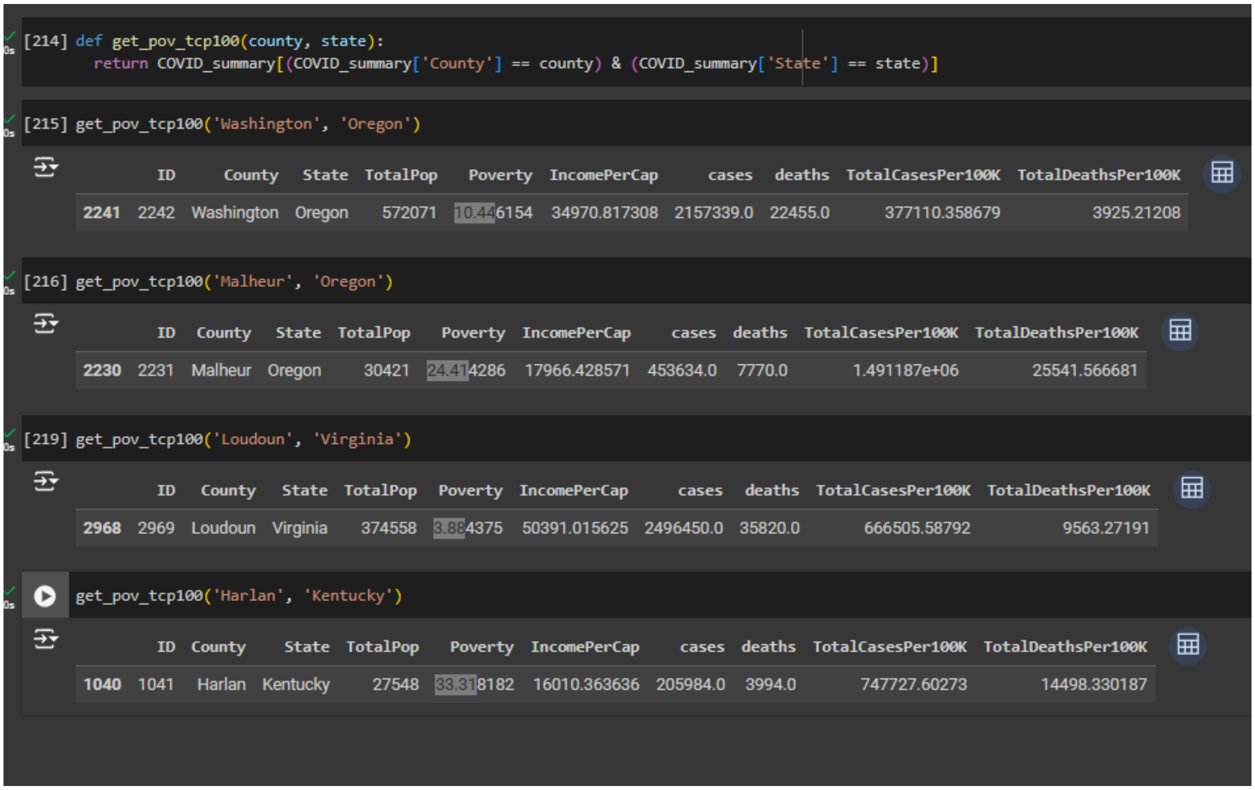


```
total_covid_data = COVID_monthly.groupby('ID').agg({
    'cases': 'sum',
    'deaths': 'sum'
})

COVID_summary = pd.merge(county_info, total_covid_data, how='left', left_on='ID', right_on='ID')
COVID_summary['TotalCasesPer100K'] = (COVID_summary['cases'] / COVID_summary['TotalPop']) * 100000
COVID_summary['TotalDeathsPer100K'] = (COVID_summary['deaths'] / COVID_summary['TotalPop']) * 100000

COVID_summary = COVID_summary[['ID', 'County', 'State', 'TotalPop', 'Poverty', 'IncomePerCap', 'cases', 'deaths', 'TotalCasesPer100K', 'TotalDeathsPer100K']]
COVID_summary.rename(columns={
    'IncomePerCap': 'PerCapitaIncome',
    'cases': 'TotalCases',
    'deaths': 'TotalDeaths'
})

COVID_summary.head()
```

```
   ID  County    State  TotalPop    Poverty  IncomePerCap      cases   deaths  TotalCasesPer100K  TotalDeathsPer100K
0   1  Autauga  Alabama     55036  14.558333  26588.166667   645935.0   9042.0       1.173659e+06        16429.246312
1   2  Baldwin  Alabama    203360  12.874194  29130.709677  2003567.0  23041.0       9.852316e+05        11330.153423
2   3  Barbour  Alabama     26201  27.755556  17891.666667   268771.0   4077.0       1.025804e+06        15560.474791
3   4     Bibb  Alabama     22580  13.925000  21799.000000   261043.0   5272.0       1.156081e+06        23348.095660
4   5   Blount  Alabama     57667  16.422222  21598.444444   630106.0   8669.0       1.092663e+06        15032.861082
```

Fill the following table using data from your COVID_summary DataFrame:

| County | Poverty % | TotalCasesPer100K |
|---|---|---|
| Washington County, Oregon | 10.44 | 3925.21 |
| Malheur County, Oregon | 24.41 | 25541.56 |
| Loudoun County, Virginia | 3.88 | 9563.27 |
| Harlan County, Kentucky | 33.31 | 14498.33 |



```
[214] def get_pov_tcp100(county, state):
        return COVID_summary[(COVID_summary['County'] == county) & (COVID_summary['State'] == state)]
```

```
[215] get_pov_tcp100('Washington', 'Oregon')
```

```
        ID      County   State  TotalPop    Poverty   IncomePerCap      cases   deaths  TotalCasesPer100K  TotalDeathsPer100K
2241  2242  Washington  Oregon    572071  10.446154   34970.817308  2157339.0  22455.0      377110.358679          3925.21208
```

```
[216] get_pov_tcp100('Malheur', 'Oregon')
```

```
        ID   County   State  TotalPop    Poverty  IncomePerCap     cases  deaths  TotalCasesPer100K  TotalDeathsPer100K
2230  2231  Malheur  Oregon     30421  24.414286  17966.428571  453634.0  7770.0       1.491187e+06        25541.566681
```

```
[219] get_pov_tcp100('Loudoun', 'Virginia')
```

```
        ID   County     State  TotalPop   Poverty  IncomePerCap      cases   deaths  TotalCasesPer100K  TotalDeathsPer100K
2968  2969  Loudoun  Virginia    374558  3.884375  50391.015625  2496450.0  35820.0       666505.58792          9563.27191
```

```
get_pov_tcp100('Harlan', 'Kentucky')
```

```
        ID  County     State  TotalPop    Poverty  IncomePerCap     cases  deaths  TotalCasesPer100K  TotalDeathsPer100K
1040  1041  Harlan  Kentucky     27548  33.318182  16010.363636  205984.0  3994.0       747727.60273        14498.330187
```

# E. [SHOULD] Analysis

For each of the following, determine the strength of the correlation between each pair of variables. Compute the correlation strength by calculating the Pearson correlation coefficient R for pairs of columns in your DataFrame. For example, if you have a DataFrame df with each row representing a distinct county, and columns named ‘TotalCasesPer100K’ and ‘Poverty’, then you can compute R like this:

```
R = df[‘TotalCasesPer100K’].corr(df[‘Poverty’])
```

1. Compute the correlation coefficient for the following relationships for all Oregon counties
    a. COVID total cases vs. % population in poverty
    b. COVID total deaths vs. % population in poverty
    c. COVID total cases vs. Per Capita Income level
    d. COVID total deaths vs. Per Capita Income level

2. Across all of the counties in the entire USA
    a. COVID total cases vs. % population in poverty
    b. COVID total deaths vs. % population in poverty
    c. COVID total cases vs. Per Capita Income level
    d. COVID total deaths vs. Per Capita Income level

Fill the following table using data from your analysis. Add more rows to the table if you find additional interesting information:

| County | R value |
|---|---|
| For Oregon Counties only: correlation between % poverty and COVID cases | |
| For all counties: correlation between population and COVID cases | |
| For Oregon counties only: correlation between PerCapitaIncome and COVID deaths | |
| For all USA counties: correlation between PerCapitaIncome and COVID cases | |
| <explore at least one other correlation computation that is made possible by this integrated data set> | |

## F. [ASPIRE] Charting

For any row (in the table above) with R > 0.5 or < -0.5 display a scatter plot (see pandas scatterplot and seaborn documentation for information about how to display scatter plots from DataFrame data).

*Note that this assignment does not constitute a competent, thorough statistical analysis of the relationships between immunological data and demographic data. It is just an example of the type of work that is often required to integrate disparate data sets.*