ETL Report Guide

By: Financial Services Group 1

Beth Vander Hoek, Jerad Ipsen, Joel Garcia, Shannon Bayless

September 23, 2022

Introduction

For this project, we explored datasets on factors potentially related to poverty rates in the United States between 1995 and 2020 at both the federal and state levels. We investigated the following seven questions:

Federal Level

1. How has the national U.S. poverty level changed between 1995-2020?
2. Is there a relationship between the U.S. national poverty level and: party control of the Senate, House, and President, inflation, or federal spending allocated to social services categories?

State Level

3. What factors influence poverty rates between 1995-2020 at the state level?
4. Is state tobacco revenue, state alcohol revenue, or drug overdose rates related to state poverty levels?
5. Are state poverty levels and state education spending related?
6. Is there a relationship between state poverty levels and state social services spending?
7. Are we able to predict state poverty levels based on the factors explored in Question 3?

We needed to clean and transform our datasets so that they were in a uniform format, primed to be successfully uploaded to our SQL database. After extracting, cleaning, and transforming the data, we then were able to use it to answer these questions, using visualizations to display our findings and predictions.

Data Sources

- History, Art & Archives: United States House of Representatives(2021). “Party Government Since 1857.” https://history.house.gov/Institution/Presidents-Coinciding/Party-Government/ Accessed on 09/21/2022.

- Iowa Community Indicators Program, Iowa State University (2021). *Annual Unemployment Rates by State* [Data set]. https://www.icip.iastate.edu/tables/employment/unemployment-states. Accessed 09/21/2022.
- Kaiser Family Foundation (2022). *Drug Overdose Death Rate (per 100,000 population)* [Data set]. https://www.kff.org/other/state-indicator/drug-overdose-death-rate-per-100000-population/?currentTimeframe=20&sortModel=%7B%22colId%22:%22Location%22,%22sort%22:%22asc%22%7D Accessed 09/21/2022.
- National Conference of State Legislatures(2022). *Legislative Partisan Composition Table* [Data set]. https://www.ncsl.org/research/about-state-legislatures/partisan-composition.aspx Accessed 09/21/2022.
- Statista(2021). *Property crime rate in the United States in 2020, by state*. https://www.statista.com/statistics/232575/property-crime-rate-in-the-us-by-state/ Accessed 9/29/22
- Statista(2021). *Reported violent crime rate in the United States in 2020, by state*. https://www.statista.com/statistics/200445/reported-violent-crime-rate-in-the-us-states/ Accessed 9/29/22
- Unified Crime Reporting Statistics (2021). *State Crime: From the CORGIS Dataset Project* [Data set]. https://corgis-edu.github.io/corgis/csv/state_crime/ Accessed 09/20/2022.
- Urban Institute (2020). *State and Local Finance Data: Exploring the Census of Governments* [Data set]. https://state-local-finance-data.taxpolicycenter.org/pages.cfm Accessed 09/19/2022.
- U.S. Bureau of Labor Statistics(2022). *American Consumer Price Index: 1913 to 2022*. https://www.rateinflation.com/consumer-price-index/usa-historical-cpi/ Accessed 09/20/2022.
- U.S. Bureau of Labor Statistics(2022). *Unemployment Rates for States, 2019 Annual Averages*. https://www.bls.gov/lau/lastrk19.htm Accessed 09/27/2022.
- U.S. Bureau of Labor Statistics(2022). *Unemployment Rates for States, 2020 Annual Averages*. https://www.bls.gov/lau/lastrk20.htm Accessed 09/27/2022.

- U.S. Census Bureau(2022). *Educational Attainment Tables* [Data set]. https://www.census.gov/topics/education/educational-attainment/data/tables.html Accessed 09/30/2022
- U.S. Census Bureau (2022). *Poverty Statistics: CPS & SAIPE (Time Series: various years)* [Data set]. https://www.census.gov/data/developers/data-sets/Poverty-Statistics.html Accessed 09/20/2022.
- White House Office of Management and Budget(2021). *Table 14.4–Total Government Expenditures by Major Category of Expenditure: 1948-2021*[Data set]. https://www.whitehouse.gov/omb/budget/historical-tables/ Accessed 09/21/2022.

Extraction

A majority of the data we collected came in the form or either a downloadable CSV or Excel file. Some data needed to be scraped from websites using python's beautifulsoup library. These data sets included data on federal party control of government, US poverty rates, and data on the consumer price index used to adjust any monetary data for inflation. We also used the Census Bureau's API to extract poverty related data.

Extracting CPI Data

We used the "Historical CPI for United States of America" website to extract the annual CPI (Consumer Price Index) from 1995 to 2020.

1. Use Beautiful Soup to access the primary table containing monthly and yearly CPI's.
2. Use a for loop to access each row in the table as well as extract each row year and annual CPI. Store this information in a dictionary.
3. Append the dictionaries to a list named cpis.
4. Convert the cpis list into a pandas DataFrame.

Extracting Federal Party Control Data

This data was extracted from the "History, Art & Archives" website.

1. Use Beautiful Soup to access the primary table containing year/congress number and House majority, Senate majority, and President party/name.
2. Use a for loop to extract each row of information and store it in a list.
3. Append the lists to another list named all_rows.
4. Convert all_rows into a pandas DataFrame.

Extracting Federal Spending Data

This data was pulled from the "White House Office of Management and Budget" website where we used *Table 14.4* from the Historical Tables Section.

1. Download the provided excel file
2. Using Python, read excel file into a pandas DataFrame

Extracting the State Party Control Data

We downloaded CSV files for this dataset from the State Partisan Composition page on the National Conference for State Legislature's website.

1. Download the timeline files "2002-2014 Partisan Composition timeline" and "1990-2000 Partisan Composition timeline", as well as the year files for 2015 through 2020.
2. Upload the downloaded CSVs to the container's directory named state-party, ready to be transformed.

Extracting the State Poverty Data

We used the Census Bureau's API to extract poverty data for each state for every year from 1995 to 2021.

1. Request an API key following this link and clicking "Request a Key" on the right hand side. Follow any prompts.

2. Using python, inside a for loop, request the following url and convert response to a dictionary. Append this dictionary to a list called poverty_list. This list will contain dictionaries from each year. https://api.census.gov/data/timeseries/poverty/saipe?get=NAME,SAEPOVRTALL_PT&for=state:*&time={**i**}&key={**apikey**} where **i** is a year between 1995 and 2021 and **apikey** is your API key.
3. Using another nested for loop, create a list of individual dictionaries that correspond to each row in the DataFrame. So for each element j in each element i in poverty_list, append j to a list called poverty_full.
4. Create a Spark DataFrame from poverty_full with column names 'State', 'PovertyPercent', 'Year', and 'StateCode'.
5. Drop the 'StateCode' column.
6. Export the DataFrame to a csv called state-poverty.csv.

Extracting the US Poverty Data

We acquired the poverty data from the Census Bureau's API in a similar way. We used the same API key as the State Poverty API request. The extraction steps are very similar.

1. Using python, inside a for loop, request the following url and convert response to a dictionary. Append this dictionary to a list called us_poverty_list. This list will contain dictionaries from each year. https://api.census.gov/data/timeseries/poverty/saipe?get=NAME,SAEPOVRTALL_PT&for=US&time={**i**}&key={**apikey**} where **i** is a year between 1995 and 2021 and **apikey** is your API key.
2. Using another nested for loop, create a list of individual dictionaries that correspond to each row in the DataFrame. So for each element j in each element i in us_poverty_list, append j to a list called us_poverty_full.
3. Create a Spark DataFrame from us_poverty_full with column names 'Geo', 'PovertyPercent', 'Year', and 'StateCode'.
4. Drop the 'Geo' and 'StateCode' column.
5. Export the DataFrame to a csv called us-poverty.csv.

Extracting the State Finance Data

Using a webpage from the tax policy center, we compiled a large dataset about different financial information for each state. This site compiles data gathered by the Census Bureau.

1. Navigate to this webpage of the Tax Policy Center's site.
2. Click 'Get Started.'
3. Select 'State' for the Level of Government.
4. Remove 'United States' from 'Selected States' and move all other states from 'Available States' to 'Selected States.'
5. Select the following series
   - R03
   - R06
   - R11
   - R13
   - R14
   - R16
   - R17
   - R27
   - R63
   - R76
   - R77
   - E019
   - E021
   - E024
   - E030
   - E036
   - E055
   - E090
   - E137
   - D01
6. Select years 1995-2020.
7. Select Per capita for the units and Real (2020 dollars).
8. Click 'View Results' and export as a CSV.

Extracting the State Education Attainment Data

This dataset was downloaded as single year CSV files from the U.S. Census Bureau, both from the Census Educational Attainment tables website and other third party sources housing older Census Educational Attainment data.

1. For years 1995-1999, download the PDF from this website
   a. Convert PDF to Excel

    b. Save each Excel workbook as a separate CSV
        i. Named Education199596, etc (2 years on each)
2. For years 2000, 2005, click on download Excel button top right corner here
    a. Save as CSV named Education200005
3. For years 2001-2004, 2006, 2009, click on Table 13 under Education Attainment Tables link on Census page
    a. Save Excel as CSV file for each year Education2001, etc.
4. For years 2007-2008, download Excel from this website
    a. Save as CSV named Education200708
5. For 2010-2020, download as CSV from Census Data explorer website
    a. For all states and outlying islands, remove Margin of error
    b. Name Education2010, etc.
6. Upload all CSVs to the finance-capstone-group1 container's directory named education-attainment

Extracting the State Crime Data

This dataset was extracted from the CORGIS and Statista websites.

1. Access the CORGIS website using the link provided in the Data Sources section of this report.
2. Locate the Download header.
3. Download the provided csv file that is under that section.
4. To obtain property crime data for 2020 Proceed to Statista's website.
5. Locate the Download header.
6. Download the provided XLS file that is under that section.
7. Obtaining the 2020 violent crime data follows similarly to collecting the property crime data.

Extracting the State Unemployment Data

This dataset was extracted from the Iowa State University's website Iowa Community Indicators Program. Additional tables were pulled from the U.S. Bureau of Labor Statistics.

1. Access the Iowa State University's website using the link provided in the Data Sources section of this report.
2. Under the Annual Unemployment Rates by State heading, locate a section titled Additional Data.
3. Download the provided excel file under the '1980-current' hyperlink.
4. Use Beautiful Soup to access the primary tables for the 2019 state unemployment data on the U.S. Bureau of Labor Statistics website.
5. Use a for loop to access each row in the table as well as extract each row state and unemployment percentage. Store this information in a list.
6. Append the lists to a new list named "total_rows".
7. Convert the list into a pandas DataFrame.
8. Extracting the 2020 state unemployment data follows similarly to the 2019 data.

Extracting the Overdose Rates Data:

This dataset was extracted from the Kaiser Family Foundation website.

1. Access the website using the link provided in the Data Sources section of this report.
2. Locate the TimeFrame section on the left and click on the year 1999.
3. Locate the tools section on the right.
4. Click the download icon to get the file in csv format.
5. Repeat steps 2 through 4 by cycling through each year until you get all the files. There will be 22 total files.

Transformation

All untransformed data is in an Azure container named finance-capstone-group1. Before doing any transformation, create a mount point called ‘/mnt/finance1/masterin’ that links to this container. That way we can read the CSVs into our databrick.

<u>Transforming CPI Data</u>

1. Drop years not in the range of 1995 to 2020 from the DataFrame.
2. Rename columns “year_cpi” and “cpi” to “Year” and “Cpi” respectively.
3. Export the DataFrame to the finance-capstone-group1 container as a csv called cpi.csv.

<u>Transforming Federal Party Control Data</u>

1. Convert each row of data into a string to be easily accessible.
2. Replace all ending tags with empty strings.
3. Split each row by “<td>” tag strings, resulting in each entry of each row becoming a list item for a list of that row’s entries.
4. Filter the year range column to only include the year range and not the congress number.
5. Split the president column by spaces to include only the party of the president.
6. Remove strings “\xa0”, “<sup>3</sup>”, and “<sup>13</sup>” from the senate majority column.
7. Change values in the senate majority column that include “Republicans / Democrats” to “Split.”
8. Remove strings “sup>6</sup” and “<sup>8</sup>” from the House majority column.
9. Replace “Democrats” and “Republicans” in the House Senate majority, and President columns to “Dem” and “Rep” respectively.
10. Split each row into two according to the Year Range column where Year Range is replaced by individual years.
    - First new row of each old row has year = {first year in year range}

    - Second new row of each old row has year = 1 + {first year in year range}
11. Reduce to only entries indexed on the interval 138 to 164 as to only use years from 1995 to 2020.
12. Create new column names titled “Year”, “House Majority”, “Senate Majority”, and “President”.
13. Export the DataFrame to the finance-capstone-group1 container as a csv called federal-party-control.csv.

Transforming Federal Spending Data

1. Read in the FederalGovtSpending.csv file from the finance-capstone-group1 container.
2. Replace column header “Table 14.4 - TOTAL GOVERNMENT EXPENDITURES BY MAJOR CATEGORY OF EXPENDITURE: 1948 - 2021” with “Year”.
3. Rename the columns using the following table as a guide.

Original Column Name	New Column Name
_c1	Total
_c2	Defense and International
_c3	Net Interest
_c4	Social Security and Medicare
_c5	Other Payments for Individuals
_c6	Other Federal
_c7	State and Local

4. Drop all rows with years not in the range of 1995 to 2020.

5. Merge this DataFrame with the previously created CPI DataFrame on the “Year” column.
6. Replace all commas with empty strings for all entries.
7. Turn each column type to float.
8. Adjust each entry in the DataFrame (except values under “Year” and “Cpi” columns) for inflation using a for loop and the following equation:

   $$New\,Value \ = \ Old\,Value \ * \ 258.811/CPI$$

   Where *CPI* is for the year of the *Old Value* and *258.811* is the CPI for 2020.

9. Drop the column for CPI.
10. Round each entry of the DataFrame to 3 decimal places.
11. Export the DataFrame to the finance-capstone-group1 container as a csv called federal-spending.csv.

<u>Transforming State Party Control</u>

1. Read in statelegiscontrol_1990_2000.csv file from the container as a Spark DataFrame named df_1990to2000 with header = True using the mount point.
   a. Drop columns ‘1990’ and ‘1992’ and change DataFrame name to df_1994to2000
   b. Add new columns for years ‘1995’, ‘1997’, ‘1999’, ‘2001’ as duplicates of current columns
      i. ‘1995’ = df_1994to2000[‘1994’]
      ii. ‘1997’ = df_1994to2000[‘1996’]
      iii. ‘1999’ = df_1994to2000[‘1998’]
      iv. ‘2001’ = df_1994to2000[‘2000’]
      v. Rename DataFrame df_1994to2001
2. Read in statelegiscontrol_2002_2014.csv file from the container as a Spark DataFrame named df_2002to2014 with header = True using mount point.
   a. Add new columns for years ‘2003’, ‘2005’, ‘2007’, ‘2009’ , ‘2011’, ‘2013’, 2015’ as duplicates of current columns

        i. '2003' = df_2002to2014['2002']
        ii. '2005' = df_2002to2014['2004']
        iii. '2007' = df_2002to2014['2006']
        iv. '2009' = df_2002to2014['2008']
        v. '2011' = df_2002to2014['2010']
        vi. '2013' = df_2002to2014['2012']
        vii. '2015' = df_2002to2014['2014 Pre-election']
        viii. Rename DataFrame df_2002to2015
    b. Rename column '2014 Pre-election' as '2014'
    c. Reorder columns to be 'State' then '2002' in ascending order to '2015'
3. Inner join df_1995to2001 and df_2002to2015 on the 'State' column.
    a. Rename DataFrame df_1995to2015
4. Read in the StateLegis_Control_2016.csv file as a Spark DataFrame named df_2016 from the container.
    a. Rename the columns 'Legis.' to '2016' and 'STATE' to 'State'.
    b. Remove rows that have values 'Control' or 'Party' in the 'State' column.
    c. Add a new column '2017' as a duplicate of column '2016'.
5. Inner join df_1995to2015 and df_2016 on the 'State' column.
    a. Rename DataFrame as df_1995to2017
6. Read in the State_Legis_Control_2018.csv file from the container as a Spark DataFrame named df_2018.
    a. Drop the columns 'Gov.' and 'State'.
    b. Rename the columns '_c0' to 'State' and 'Legis.' to '2018'.
    c. Remove any rows where the value in the 'State' column is 'STATE'.
    d. From pySpark.sql.functions, import the translate library.
    e. Use translate to change the * character in the 'State' and '2018' columns to a blank space.

    ```
    df_2018 = df_2018.withColumn('State', translate('State', '*', ''))
    df_2018 = df_2018.withColumn('2018', translate('2018', '*', ''))
    ```

    f. Add a new column called '2019' as a duplicate of the '2018' column.
7. Inner join df_1995to2017 with df_2018 on the 'State' column.

    a. Rename the DataFrame to df_1995to2019.
8. Read in the State_Legis_Control_2020.csv file as a Spark DataFrame named df_2020.
    a. Drop the columns named ‘Gov.’ and ‘State’.
    b. Rename the column ‘_c0’ to ‘State’ and column ‘Legis.’ to ‘2020’.
    c. Remove all rows where the ‘State’ column equals ‘STATE’.
9. Inner join df_1995to2019 with df_2020 on the ‘State’ column.
    a. Rename the DataFrame df_1995to2020.
10. Convert df_1995to2020 to a pandas DataFrame.
11. Reformat the DataFrame to move column names into column values:
    a. Create an empty list named dfs.
    b. Create a for loop for i in range(1995, 2021) to add df_1995to2020[i] to the dfs list.
    c. Create a series from the column ‘State’ and name it ‘state’.
    d. Use list comprehension to make a list of years from 1995 to 2020. Name the list ‘years’.
    e. Create an empty list named large_list.
    f. Use this for loop to make large_list a list of lists where each element is a list containing a State, Year, and Party.

```
large_list = []
for i in range(len(state)):
    for j in range(len(years)):
        list = []
        list.append(state[i])
        list.append(years[j])
        list.append(dfs[j][i])
        large_list.append(list)
```

12. Convert large_list list into a pandas DataFrame named df_stateparty.
    a. Replace ‘Divided’ values with ‘Split’, ‘NA’ values with ‘N/A’, and ‘*Dem’ values with ‘Dem’ for the whole DataFrame.
13. Convert df_stateparty into a Spark DataFrame.

14. Rename column ‘0’ to ‘State’, ‘1’ to ‘Year’, and ‘2’ to ‘Party’.
15. Export the DataFrame to the finance-capstone-group1 container as a csv called state-party.csv.

<u>Transforming State Poverty Data</u>

All state level poverty data was already in a usable form. No transformation was necessary. However, this dataset was sent through a Kafka Producer and Consumer in order to be migrated to the proper storage container.

1. Read in the poverty-rates.csv file from the finance-capstone-group1 storage container. Create a Spark DataFrame called poverty.
2. Create a new kafka topic called ‘poverty-data.’
3. Every 5 seconds, produce a message which contains a dictionary with all columns from the poverty-data DataFrame. Each message represents a single row in the DataFrame.
4. Once all messages have been produced, create a consumer to consume the messages. The messages will be collected in a list called kafkaListDictionaries.
5. Convert kafkaListDictionaries into a Spark DataFrame.
6. Export the DataFrame to the finance-capstone-group1 container as a csv called poverty-rates.csv.

<u>Transforming US Poverty Data</u>

All national poverty data was already in a usable form. No transformation was necessary.

<u>Transforming State Finance Data</u>

Not much transformation needed to happen to the State Finance Data.

1. Read in the tax-expenses.csv file into a Spark DataFrame.
2. Rename the columns to make them easier to use. Use the following table as a guide when renaming the columns.

Original Column Name	New Column Name
(R03) General Revenue	GeneralRevenue

(R06) Property Tax (T01)	PropertyTax
(R11) Alcoholic Beverage Tax (T10)	AlcoholTax
(R13) Insurance Premium Tax (T12)	InsurancePremiumTax
(R14) Motor Fuels Tax (T13)	MotorFuelsTax
(R16) Public Utility Tax (T15)	PublicUtilityTax
(R17) Tobacco Tax (T16)	TobaccoTax
(R27) Individual Income Tax (T40)	IndividualIncomeTax
(R63) Liquor Stores Revenue (A90)	LiquorStoreRevenue
(R76) Total Unemp Rev	UnemploymentRevenue
(R77) Unemp-Payroll Tax (Y01)	UnempPayrollTax
(E019) Police & Fire Protection-Dir Exp	PoliceFireExpenses
(E021) Total Correct-Dir Exp	TotalCorrectionalExpenses
(E024) Total Educ-Direct Exp	TotalEducationExpenses
(E036) Educational Assistance (E19)	EducationAssistanceExpenses
(E055) Health-Direct Expend	HealthExpenses
(E090) Public Welf-Direct Exp	PublicWelfareExpenses
(E137) Unemp Comp-Total Exp	UnemploymentExpenses
(D01) Total Debt Outstanding	TotalDebt

3. Remove all commas and dollar signs from the data.

4. Export the DataFrame to the finance-capstone-group1 container as a csv called state-finances.csv.

<u>Transforming State Education Attainment Data</u>

1. Create mount to finance-capstone-group1 container
2. Read in Education1995.csv using mount from education-attainment directory as Spark dataframe named df_1995 with header = True
    a. Drop all na values
    b. Filter so column '_c1' does not equal 'Total'
    c. Drop columns '_c1' '_c3' '_c5' '_c6' '_c7' '_c8' '_c9' '_c10'
    d. Rename columns 'Educational Attainment of the Population 25 Years and Over, By State,' as 'State' and '_c2' as 'HS1995' and '_c4' as 'BA1995'
3. Read in Education199697.csv using mount from education-attainment directory as Spark dataframe named df_199697 with header = True
    a. Drop all na values
    b. Filter so column '_c1' does not equal 'Total'
    c. Drop columns '_c1' '_c3' '_c5' '_c6' '_c8' '_c10'
    d. Rename columns 'Educational Attainment of the Population 25 Years and Over, By State,' as 'State', '_c2' as 'HS1997', '_c4' as 'BA1997', '_c7' as 'HS1996' and '_c9' as 'BA1996'
4. Read in Education199899.csv using mount from education-attainment directory as Spark dataframe named df_199899 with header = True
    a. Drop all na values
    b. Filter so column '_c1' does not equal 'Total'
    c. Drop columns '_c1' '_c3' '_c5' '_c6' '_c8' '_c10'
    d. Rename columns 'Educational Attainment of the Population 25 Years and Over, By State,' as 'State', '_c2' as 'HS1999', '_c4' as 'BA1999', '_c7' as 'HS1998' and '_c9' as 'BA1998'
5. Read in Education200005.csv using mount from education-attainment directory as Spark dataframe named df_200005 with header = True

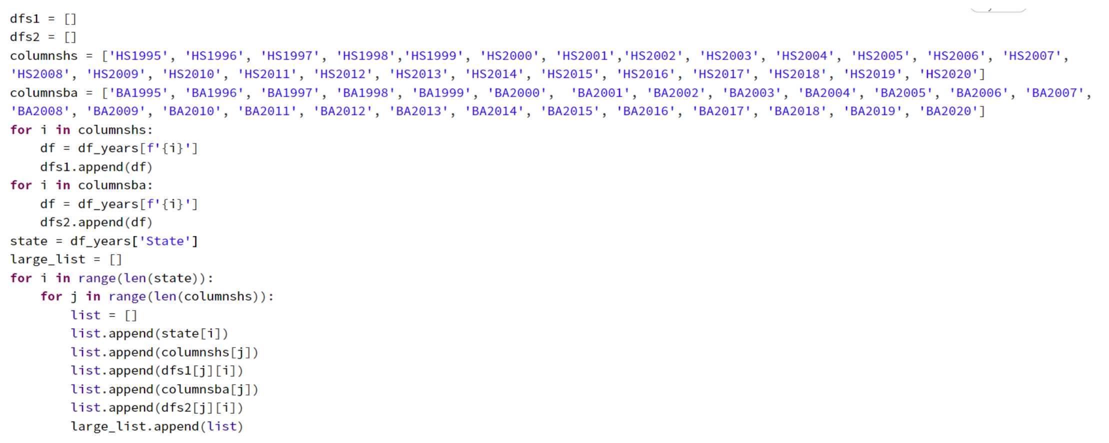
   a. Drop columns '_c1', '_c2', '_c3', '_c4', '_c5', '_c8', '_c9', '_c10', '_c11', '_c13', '_c15', '_c16', '_c17', '_c18', '_c19', '_c20', '_c21', '_c22', '_c23', '_c24', '_c25', '_c26', '_c27'
   b. Rename columns ‘Table 11. Educational attainment of persons 18 years old and over, by state: 2000 and 2005’ as ‘State’, ‘_c6’ as ‘HS2000’, ‘_c7’ as ‘BA1999’, ‘_c12’ as ‘HS2005’ and ‘_c14’ as ‘BA2005’
   c. Drop na values from subset ‘State’ and ‘HS2000’ columns
   d. Filter so ‘State’ column does not equal ‘State’, ‘1’, or ‘United States ……..’
   e. Import translate from pyspark.sql.functions
   f. Translate ‘State’ column so ‘ ....’ is replaced with ‘’ (aka remove)
   g. Import regexp_replace from pyspark.sql.functions
   h. Replace all state names in the ‘State’ column with two words to have space between words (ex. ‘NewJersey’ to ‘New Jersey’)
6. Read in Education2001 using mount from education-attainment directory as Spark dataframe named df_2001 with header = True
   a. Drop na values
   b. Drop columns ‘_c1’ ‘_c3’ ‘_c5’
   c. Rename columns ‘table with row headers in column A and column headers in row 5 and repeated in row 41’ as ‘State’, ‘_c2’ as ‘HS2001’ and ‘_c4’ as ‘BA2001’
7. Read in Education2002 using mount from education-attainment directory as Spark dataframe named df_2002 with header = True
   a. Drop columns ‘_c1’ ‘_c3’ ‘_c5’ ‘_c6’
   b. Drop na values
   c. Rename columns ‘table with row headers in column A and column headers in row 5 and repeated in row 41’ as ‘State’, ‘_c2’ as ‘HS2002’ and ‘_c4’ as ‘BA2002’
8. Read in Education2003 using mount from education-attainment directory as Spark dataframe named df_2003 with header = True
   a. Drop columns ‘_c1’ ‘_c3’ ‘_c5’

   b. Rename columns ‘Table with row headers in column A and column headers in row 5’ as ‘State’, ‘_c2’ as ‘HS2003’ and ‘_c4’ as ‘BA2003’
   c. Drop na values
9. Read in Education2004 using mount from education-attainment directory as Spark dataframe named df_2003 with header = True
   a. Drop columns ‘_c1’ ‘_c3’ ‘_c5’
   b. Rename columns ‘Table with row headers in column A and column headers in row 5’ as ‘State’, ‘_c2’ as ‘HS2004’ and ‘_c4’ as ‘BA2004’
   c. Drop na values
10. Read in Education2006 using mount from education-attainment directory as Spark dataframe named df_2006 with header = True
   a. Drop columns ‘_c1’ ‘_c3’ ‘_c5’
   b. Rename columns ‘Table with row headers in column A and column headers in rows 5 and 6’ as ‘State’, ‘_c2’ as ‘HS2006’ and ‘_c4’ as ‘BA2006’
   c. Drop na values
   d. Filter so ‘State’ column does not equal ‘State’
11. Read in Education200708 using mount from education-attainment directory as Spark dataframe named df_200708 with header = True
   a. Drop columns ‘_c1', '_c2', '_c3', '_c4', '_c5', '_c6', '_c7', '_c8', '_c9', '_c10', '_c11', '_c12', '_c13', '_c14', '_c15', '_c16', '_c18', '_c20', '_c21', '_c22', '_c23', '_c24', '_c25', '_c26', '_c27', '_c28', '_c29', '_c30', '_c31', '_c32’
   b. Rename columns ‘Table 11. Educational attainment of persons 18 years old and over, by state: 2000 and 2006-08’ as ‘State’, ‘_c17’ as ‘HS2007’ and ‘_c19’ as ‘BA2007’
   c. Drop na values from ‘State’ and ‘HS2007’ columns
   d. Filter so ‘State’ column does not equal ‘State’, ‘1’, or ‘United States ……..’
   e. Translate ‘State’ column so ‘ ....’ becomes ‘’ (aka remove)
   f. Replace all state names in the ‘State’ column with two words to have space between words (ex. ‘NewJersey’ to ‘New Jersey’)

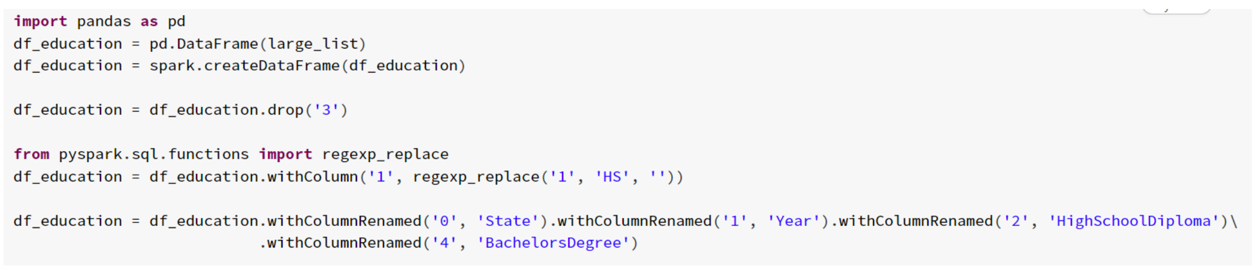
    g. Add new column ‘HS2008’as copy of ‘HS2007’ and ‘BA2008’ as copy of ‘BA2007’

12. Read in Education2009 using mount from education-attainment directory as Spark dataframe named df_2009 with header = True
    a. Drop columns ‘_c2', '_c3', '_c4', '_c5', '_c6', '_c8', '_c9', '_c10', '_c11', '_c12'
    b. Rename columns ‘Area’ as ‘State’, ‘High school or more education2’ as ‘HS2009’ and ‘Bachelor’s degree or more’ as ‘BA2009’
    c. Drop all na values
    d. Translate ‘State’ column so ‘ ....’ becomes ‘’ (aka remove)
    e. Filter so ‘State’ column does not equal ‘UnitedStates’ ‘Northeast’ ‘Midwest’ ‘South’ or ‘West’
    f. Replace all state names in the ‘State’ column with two words to have space between words (ex. ‘NewJersey’ to ‘New Jersey’)
13. Read in Education2010 using mount from education-attainment directory as Spark dataframe named df_2010 with header = True
    a. Filter so ‘Label (Grouping)’ column only equals ‘Percent high school graduate or higher’ or ‘Percent bachelor\’s degree or higher’
    b. Create a list of column names to keep all 50 states plus District of Columbia following this naming format: ‘Iowa!!Total!!Estimate’
    c. Set dataframe equal to only those columns of original dataframe
    d. Convert Spark dataframe to Pandas dataframe
    e. Use lambda function to replace ‘%’ with ‘’ in all columns
    f. Create list of column names and remove ‘Label (Grouping)’ from list
    g. Transpose rows and columns
    h. Use first row as new headers
    i. Create new ‘State’ column equal to list of column names made above
    j. Convert dataframe to Spark dataframe
    k. Replace ‘!!Total!!Estimate’ with ‘’ in ‘State’ column
    l. Rename column ‘0’ as ‘HS2010’ and ‘1’ as ‘BA2010’
    m. Reorder columns to be ‘State’ ‘HS2010’ ‘BA2010’

14. Repeat above process for Education2011, Education2012, Education2013, Education2014, changing HS, BA column and df names for each year
15. Repeat above process for Education2015 and Education2016 except create list of column names to keep all 50 states plus District of Columbia following this naming format: ‘Iowa!!Percent!!Estimate’ (replace Total with Percent)
16. Repeat same process for Education2017-2020 as Education2015/2016 except keep only rows where ‘Alabama!!Percent!!Estimate’ is equal to correct HS and BA percentages for that year (check CSV for values for each year)
    a. Ex. 2017 keep rows equal to ‘85.3%’ and ‘24.5’
17. Convert all Spark dataframes to Pandas dataframes
18. Add new columns ‘HS’ and ‘BA’ from each dataframe to df_1995 in year order
    a. Rename df_1995 as df_years
19. Follow the code below to transform table to format that matches SQL database

20.


```
dfs1 = []
dfs2 = []
columnshs = ['HS1995', 'HS1996', 'HS1997', 'HS1998','HS1999', 'HS2000', 'HS2001','HS2002', 'HS2003', 'HS2004', 'HS2005', 'HS2006', 'HS2007',
'HS2008', 'HS2009', 'HS2010', 'HS2011', 'HS2012', 'HS2013', 'HS2014', 'HS2015', 'HS2016', 'HS2017', 'HS2018', 'HS2019', 'HS2020']
columnsba = ['BA1995', 'BA1996', 'BA1997', 'BA1998', 'BA1999', 'BA2000',  'BA2001', 'BA2002', 'BA2003', 'BA2004', 'BA2005', 'BA2006', 'BA2007',
'BA2008', 'BA2009', 'BA2010', 'BA2011', 'BA2012', 'BA2013', 'BA2014', 'BA2015', 'BA2016', 'BA2017', 'BA2018', 'BA2019', 'BA2020']
for i in columnshs:
    df = df_years[f'{i}']
    dfs1.append(df)
for i in columnsba:
    df = df_years[f'{i}']
    dfs2.append(df)
state = df_years['State']
large_list = []
for i in range(len(state)):
    for j in range(len(columnshs)):
        list = []
        list.append(state[i])
        list.append(columnshs[j])
        list.append(dfs1[j][i])
        list.append(columnsba[j])
        list.append(dfs2[j][i])
        large_list.append(list)
```

21.


```
import pandas as pd
df_education = pd.DataFrame(large_list)
df_education = spark.createDataFrame(df_education)

df_education = df_education.drop('3')

from pyspark.sql.functions import regexp_replace
df_education = df_education.withColumn('1', regexp_replace('1', 'HS', ''))

df_education = df_education.withColumnRenamed('0', 'State').withColumnRenamed('1', 'Year').withColumnRenamed('2', 'HighSchoolDiploma')\
                              .withColumnRenamed('4', 'BachelorsDegree')
```

22. Create mount point to container finance-capstone-group1-consumer
23. Write Spark dataframe as CSV to container using mount point where header = True and mode = overwrite

Transforming State Crime Data

1. Read in the state-crime.csv file as a Spark DataFrame.
2. Convert the Spark DataFrame into a pandas DataFrame
3. Filter out the years column to reflect data from 1995 to 2019
4. Drop all columns except ‘State’, ‘Year’, Population, ‘Data.Rates.Property.All’, and ‘Data.Rates.Violent.All’.
5. Change the names of the ‘Data.Rates.Property.All’, and ‘Data.Rates.Violent.All’ columns to ‘Property Crime Rates’ and ‘Violent Crime Rates.’
6. Change the data type for the ‘Year’ and ‘Population’ columns to integer.
7. Change the data type for the ‘Property Crime Rates’ and ‘Violent Crime Rates’ columns to float.
8. Read in the violent-crime.xlsx file as a pandas DataFrame.
9. Drop the “Unnamed: 0” column from the violent crime DataFrame.
10. Drop the first 4 rows of the violent crime DataFrame.
11. Rename columns "Unnamed: 1" and "Unnamed: 2" to "State" and "Violent Crime Rates" respectively.
12. Read in the property-crime.xlsx file as a pandas DataFrame.
13. Drop the “Unnamed: 0” column from the property crime DataFrame.
14. Drop the first 4 rows of the property crime DataFrame.
15. Rename columns "Unnamed: 1" and "Unnamed: 2" to "State" and "Property Crime Rates" respectively.
16. Merge the property crime and violent crime DataFrames on the state column.
17. Add a year column with the float value of 2020 to the merged DataFrame.
18. Concatenate the merged DataFrame with the state crime DataFrame.
19. Reset the index of the new DataFrame.
20. Convert the pandas DataFrame back into a Spark DataFrame.
21. Export the DataFrame to the finance-capstone-group1 container as a csv called state-crime.csv.

Transforming State Unemployment Data:

1. Read in the state-unemployment.csv file as a Spark DataFrame with header='True' and skiprows=5.
2. Convert the Spark DataFrame into a pandas DataFrame.
3. Every year has its own column. Filter out the columns to only have the "Area" column along with the 1995 through 2018 columns.
4. Convert the data types of the 1995 to 2018 columns to float.
5. Rename the column "Area" to "State"
6. Drop NaN values. There are some notes made about the dataset itself that got incorporated under the State column. This step gets rid of these rows.
7. Create a list that will take in the state column, the percentages, and the years for each row with the following code:

```
dfs = []
for i in range(1995, 2019):
    df = new_df[f'{i}']
    dfs.append(df)
state = new_df['State']
def createList(r1, r2):
    return [item for item in range(r1, r2+1)]
years = createList(1995, 2018)
large_list = []
for i in range(len(state)):
    for j in range(len(years)):
        list = []
        list.append(state[i])
        list.append(years[j])
        list.append(dfs[j][i])
        large_list.append(list)
large_list
```

This has to be done to create a new DataFrame to have a proper Year column.

8. Load large_list, created in the previous step, into a pandas DataFrame.
9. Introduce pandas DataFrames created for the 2019 and 2020 state unemployment data in the extraction stage.
10. Add a year column for each and insert their respective year.
11. Concatenate these three DataFrames into one.
12. Reset the index for the new concatenated DataFrame.
13. Convert the pandas DataFrame into a Spark DataFrame.
14. Rename the columns in the Spark DataFrame:
    - "0" as "State"
    - "1" as "Year"
    - "2" as "Unemployment Percentage Rate"
15. Export the DataFrame to the finance-capstone-group1 container as a csv called state-unemployment.csv.

<u>Transforming Overdose Rates Data</u>

The Drug Overdose Death Rate datasets were taken from KFF that includes drug overdose deaths separated by location within the US from years ranging from 1999 to 2020. We used the following steps to transform the data into a usable format.

1. Load in the multiple csv files within the dataset with this:

```
df = Spark.read.format("csv").load("/mnt/yourname/overdose-in/data",
header=True, inferSchema='True',skiprows=2).withColumn('FilePath',
input_file_name())
```

2. Convert the Spark DataFrame into a pandas DataFrame.
3. Rename the 'FilePath' column into a 'Year' column.
4. Use this code to convert the FilePath values into years with this code:

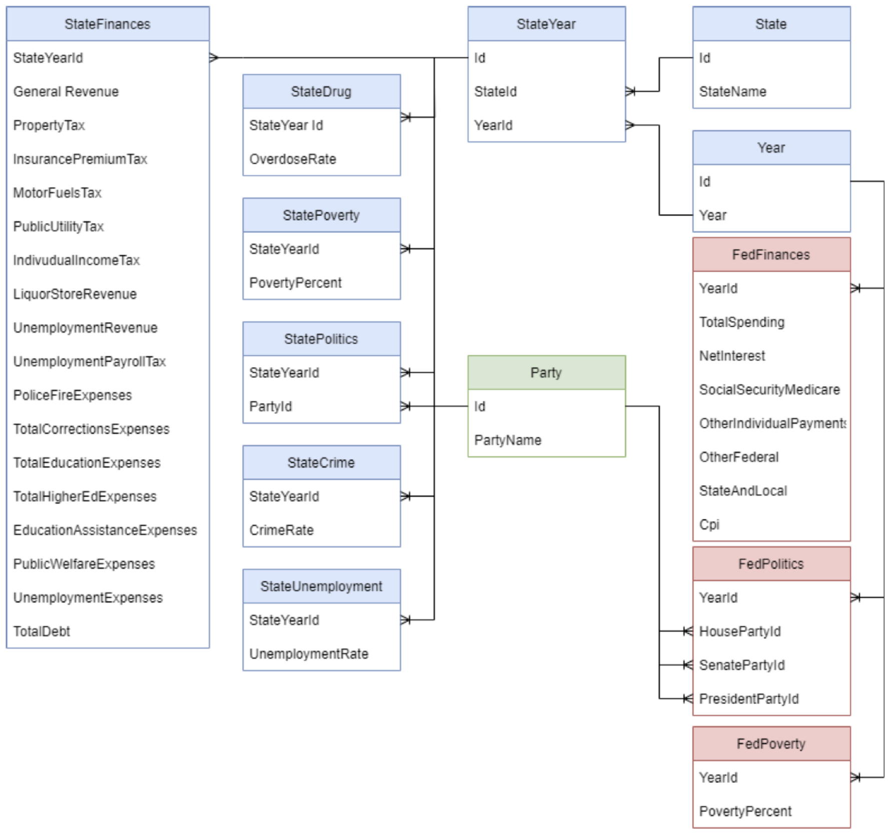
```
for i in range(1,23):
    year = 1998 + i
```

```
new_df.replace(f"dbfs:/mnt/yourname/overdose-in/data/raw_data_{i}.cs
v", f"{year}",inplace=True)
```

5. Rename the 'Location' column to 'State'.
6. Drop null values.
7. Convert the Year column's data type into integer.
8. Sort the values by State in order to have everything in alphabetical order.
9. Convert the pandas DataFrame back into a Spark DataFrame.
10. Export the DataFrame to the finance-capstone-group1 container as a csv called state-od.csv.

Load

After transforming individual datasets, we manipulated the tables to reflect our entity relationship diagram before loading everything into a SQL database.



Read in the following CSVs as a Spark DataFrame. See the table below to reference the CSVs and their corresponding DataFrame Names.

CSV Name	**DataFrame Name**
poverty-rates.csv	statePovertyTable
state-finances.csv	stateFinancesTable
state-party.csv	statePoliticsTable
state-crime.csv	stateCrimeTable
state-od.csv	stateDrugTable
state-unemployment.csv	stateUnemploymentTable
federal-party-control.csv	fedPoliticsTable
federal-spending.csv	fedFinanceTable
us-poverty-rates.csv	fedPovertyTable
cpi.csv	cpiTable

Party Table

The party table in the SQL database has columns ‘Party’ and ‘PartyId’.

1. Select the distinct values from the ‘Party’ column in the statePoliticsTable DataFrame. Store these in a DataFrame called ‘party’.
2. Create a column in the ‘party’ DataFrame called ‘PartyId’ and use the monotonically_increasing_id function to populate the column with Id numbers corresponding to the political parties.

State Table

The state table in the SQL database has columns ‘State’ and ‘StateId’.

1. Select the distinct values from the ‘State’ column in the statePovertyTable DataFrame. Store these in a DataFrame called ‘state’.
2. Create a column in the ‘state’ DataFrame called ‘StateId’ and use the monotonically_increasing_id function to populate the column with Id numbers corresponding to the states.

Year Table

The year table in the SQL database has columns ‘Year’ and ‘YearId’.

1. Select the distinct values from the ‘Year’ column in the statePovertyTable DataFrame. Store these in a DataFrame called ‘year’.
2. Create a column in the ‘year’ DataFrame called ‘YearId’ and use the monotonically_increasing_id function to populate the column with Id numbers corresponding to the years.

StateYear Table

The StateYear table in the SQL database has columns ‘StateYearId’, ’StateId’, and ‘YearId’.

1. Select the distinct values from the ‘Year’ and ‘State’ columns in the statePovertyTable DataFrame. Store these in a DataFrame called ‘stateYear’.
2. Create a column in the ‘stateYear’ DataFrame called ‘StateYearId’ and use the monotonically_increasing_id function to populate the column with Id numbers corresponding to unique combinations of states and years.
3. Using an inner join, join the stateYear and state DataFrames on the ‘State’ column. Call this joined DataFrame ‘stateYear’.
4. Using an inner join, join the stateYear and year DataFrames on the ‘Year’ column. Call this joined DataFrame ‘stateYear’.
5. Drop the ‘State’ and ‘Year’ columns from the stateYear DataFrame.

StateFinances Table

The StateFinances table has columns ‘StateYearId’ and all other financial columns in the stateFinancesTable DataFrame.

1. Join the stateFinancesTable and state DataFrames on ‘State’ using an inner join. Name the merged DataFrame stateFinances.
2. Join the stateFinances and year DataFrames on ‘Year’ using an inner join. Name the merged DataFrame stateFinances.
3. Join the stateFinances and stateYear DataFrames on ‘YearId’ and ‘StateId’ using an inner join. Name the merged DataFrame stateFinances.
4. Drop the ‘YearId’, ‘StateId’, ‘Year’, and ‘State’ columns from the stateFinances DataFrame.

<u>StateDrug Table</u>

The StateDrug table has columns ‘StateYearId’ and ‘DurgOverdoseRate’

1. Join the stateDrugTable and state DataFrames on ‘State’ using an inner join. Name the merged DataFrame stateDrug.
2. Join the stateDrug and year DataFrames on ‘Year’ using an inner join. Name the merged DataFrame stateDrug.
3. Join the stateDrug and stateYear DataFrames on ‘YearId’ and ‘StateId’ using an inner join. Name the merged DataFrame stateDrug.
4. Drop the ‘YearId’, ‘StateId’, ‘Year’, and ‘State’ columns from the stateDrug DataFrame.

<u>StateEducation Table</u>

The StateEducation table has columns ‘StateYearId’, ‘HighSchoolDiploma’, and ‘BachelorsDegree’.

1. Join the stateEducationTable and state DataFrames on ‘State’ using an inner join. Name the merged DataFrame stateEducation.
2. Join the stateEducation and year DataFrames on ‘Year’ using an inner join. Name the merged DataFrame stateEducation.
3. Join the stateEducation and stateYear DataFrames on ‘YearId’ and ‘StateId’ using an inner join. Name the merged DataFrame stateEducation.
4. Drop the ‘YearId’, ‘StateId’, ‘Year’, and ‘State’ columns from the stateEducation DataFrame.

StateCrime Table

The StateDrug table has columns ‘StateYearId’, ‘PropertyCrimeRate’, and ‘ViolentCrimeRate’.

1. Join the stateCrimeTable and state DataFrames on ‘State’ using an inner join. Name the merged DataFrame stateCrime.
2. Join the stateCrime and year DataFrames on ‘Year’ using an inner join. Name the merged DataFrame stateCrime.
3. Join the stateCrime and stateYear DataFrames on ‘YearId’ and ‘StateId’ using an inner join. Name the merged DataFrame stateCrime.
4. Drop the ‘YearId’, ‘StateId’, ‘Year’, and ‘State’ columns from the stateCrime DataFrame.

StatePolitics Table

The StatePolitcs table has columns ‘StateYearId’ and ‘PartyId’.

1. Join the statePoliticsTable and state DataFrames on ‘State’ using an inner join. Name the merged DataFrame statePolitics.
2. Join the statePolitics and year DataFrames on ‘Year’ using an inner join. Name the merged DataFrame statePolitics.
3. Join the statePolitics and stateYear DataFrames on ‘YearId’ and ‘StateId’ using an inner join. Name the merged DataFrame statePolitics.
4. Join the statePolitics and party DataFrames on ‘Party’ using an inner join. Name the merged DataFrame statePolitics.
5. Drop the ‘Party’, ‘YearId’, ‘StateId’, ‘Year’, and ‘State’ columns from the statePolitics DataFrame.

StateUnemployment Table

The StateUnemployment table has columns ‘StateYearId’, ‘UnemploymentRate’.

1. Join the stateUnemploymentTable and state DataFrames on ‘State’ using an inner join. Name the merged DataFrame stateUnemp.
2. Join the stateUnemp and year DataFrames on ‘Year’ using an inner join. Name the merged DataFrame stateUnemp.

3. Join the stateUnemp and stateYear DataFrames on ‘YearId’ and ‘StateId’ using an inner join. Name the merged DataFrame stateUnemp.
4. Drop the ‘YearId’, ‘StateId’, ‘Year’, and ‘State’ columns from the stateUnemp DataFrame.

<u>FederalFinances Table</u>

The FederalFinances table has columns ‘YearId’, ‘CPI’, and all other financial columns from the fedFinancesTable DataFrame.

1. Join the fedFinancesTable and year DataFrames on ‘Year’ using an inner join. Name the merged DataFrame fedFinances.
2. Join the fedFinances and cpiTable DataFrames on ‘Year’ using an inner join. Name the merged DataFrame fedFinances.
3. Drop the ‘Year’ column from the fedFinances DataFrame.

<u>FederalPoverty Table</u>

The FederalFinances table has columns ‘YearId’ and ‘PovertyPercent’.

1. Join the fedPovertyTable and year DataFrames on ‘Year’ using an inner join. Name the merged DataFrame fedPoverty.
2. Drop the ‘Year’ and ‘Geo’ column from the fedPoverty DataFrame.

<u>FederalPolitics Table</u>

The FederalPolitics table has columns ‘YearId’, ‘HouseId’, ’SenateId’, and ‘PresidentId’

4. Join the fedPoliticsTable and year DataFrames on ‘Year’ using an inner join. Name the merged DataFrame fedPolitics.
5. Convert the fedPolitics and party DataFrames to pandas DataFrames called fedPoliticspandas and partypandas respectively.
6. For the House, Senate, and President columns, complete the following steps.
    a. In the partypandas DataFrame, rename the ‘Party’ column to either ‘BRANCH Majority’, where BRANCH is either House, Senate, or ‘President’ when working with the president column.

   b. Using an inner join, merge fedPoliticspandas and partypandas on ‘BRANCH Majority’, or ‘President’ (depending on what column we are working on).
   c. Rename the ‘PartyId’ column in the fedPoliticspandas DataFrame to ‘BRANCHId’ where BRANCH refers to the current working column.
7. Convert fedPoliticspandas back to a Spark DataFrame.
8. Drop the ‘Year’, ‘House Majority’, ‘Senate Majority’, and ‘President’ columns from the fedPolitics DataFrame.

<u>Loading into SQL Database</u>

Use the following table as a guide to match up DataFrames to their SQL tables.

DataFrame Name	SQL Table
year	dbo.Year
state	dbo.State
stateYear	dbo.StateYear
party	dbo.Party
stateFinances	dbo.StateFinances
statePoverty	dbo.StatePoverty
stateDrug	dbo.StateDrug
statePolitics	dbo.StatePolitics
stateCrime	dbo.StateCrime
stateUnemp	dbo.StateUnemployment
fedFinances	dbo.FederalFinances

fedPoverty	dbo.FederalPoverty
fedPolitics	dbo.FederalPolitics

Use the following code block as a guide for how to write each table to the SQL Database.

```
database = "databasename"

user = "username"

password  = "password"

jdbcUrl =
f"jdbc:sqlserver://gen10-data-fundamentals-22-07-sql-server.database.w
indows.net:1433;database=capstone-finance-group1;user=gen10dbadmin@gen
10-data-fundamentals-22-07-sql-server;password={password};encrypt=true
;trustServerCertificate=false;hostNameInCertificate=*.database.windows
.net;loginTimeout=30;"

DATAFRAME.write.format("jdbc") \

  .mode("overwrite") \

  .option("url", jdbcUrl) \

  .option("dbtable", "dbo.TABLENAME") \

  .option("user", user) \

  .option("password", password) \

  .save()
```

Conclusion

By extracting and transforming 9 different datasets, we successfully created tables that could be loaded into a SQL database. We are now able to make meaningful visualizations to represent the data as well as create and fit a Lasso Machine Learning model to predict state poverty levels.