Data Visualization Using Flask App

Project-2

By: Mubashira Qari

Data Selection and Sources

The purpose of data selection is to present a story from the census and education data for the year 2019. The datasets are selected from the following sources:

US county and states level 5 years census dataset through an API call.

US county and states level education attainment dataset from ‘Economic Research Services’
(USDA ERS) (LINK)

The fips code dataset downloaded from github.

Requirement Accomplished

The following specification required for the project are utilized and fulfilled:

- Python Flask - powered API
- HTML/CSS
- Javascript
- PostgreSQL database
- Leaflet - Javascript Library
- Chart.js - Javascript Library
- Plotly - Javascript graphing Library
- User-driven interaction (e.g., menus, dropdowns, and text boxes).
- A dashboard page with multiple charts that updates from the same data.
- A server that performs multiple manipulations on data in a database prior to visualization.

Data Fetching and Cleaning Steps

- The dependencies imported are: Pandas, request, Census, api_key
- In file ‘census_2019_county_apidata.ipynb’,
  the census dataset is retrieved through the API call.

- The file is saved as ‘census_us_county_output.csv’

- Next, the data cleaning is performed in the file,
  ‘data_cleaning.ipynb’

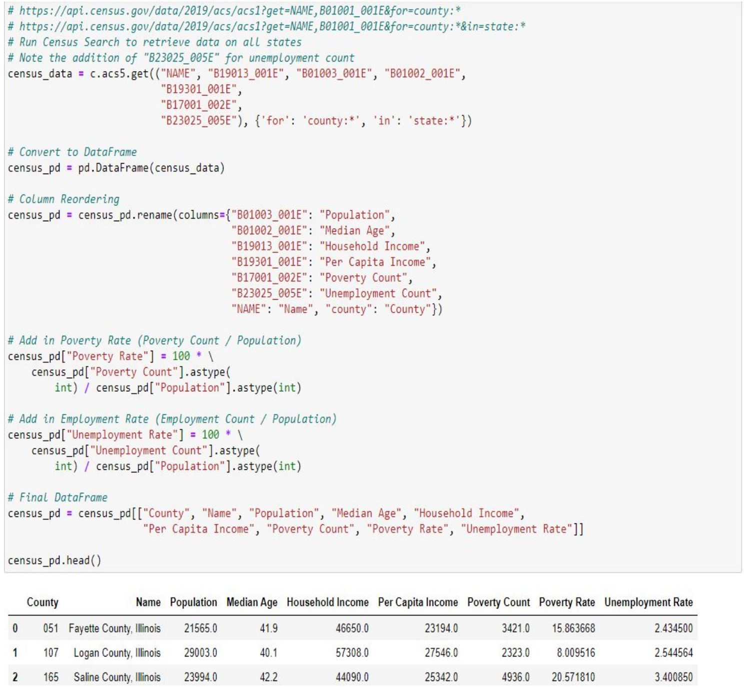
- Three csv files are imported in ‘data_cleaning.ipynb’
  for cleaning and preparing for the PostgreSQL database:

  "resources/census_us_county_output.csv"

  "resources/ers_usda_education.csv"

  "resources/county_fips.csv"

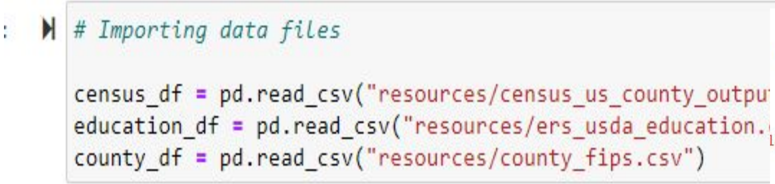
- In Leaflet file, ‘leaflet_data_cleaning.ipynb’ , data cleaning
   is performed for the leaflet map.



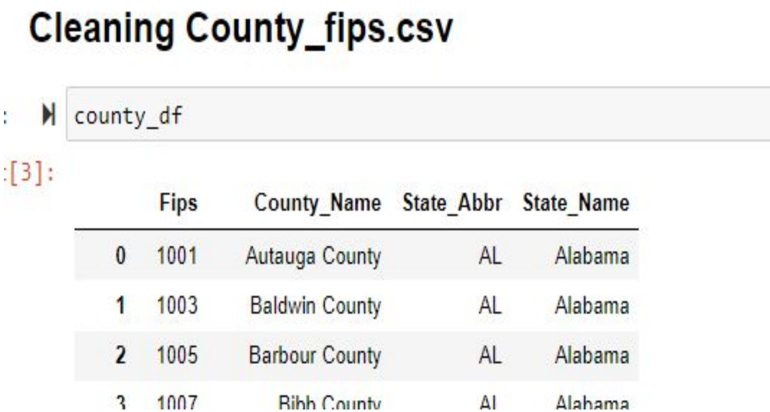
```
# https://api.census.gov/data/2019/acs/acs1?get=NAME,B01001_001E&for=county:*
# https://api.census.gov/data/2019/acs/acs1?get=NAME,B01001_001E&for=county:*&in=state:*
# Run Census Search to retrieve data on all states
# Note the addition of "B23025_005E" for unemployment count
census_data = c.acs5.get(("NAME", "B19013_001E", "B01003_001E", "B01002_001E",
                          "B19301_001E",
                          "B17001_002E",
                          "B23025_005E"), {'for': 'county:*', 'in': 'state:*'})

# Convert to DataFrame
census_pd = pd.DataFrame(census_data)

# Column Reordering
census_pd = census_pd.rename(columns={"B01003_001E": "Population",
                                      "B01002_001E": "Median Age",
                                      "B19013_001E": "Household Income",
                                      "B19301_001E": "Per Capita Income",
                                      "B17001_002E": "Poverty Count",
                                      "B23025_005E": "Unemployment Count",
                                      "NAME": "Name", "county": "County"})

# Add in Poverty Rate (Poverty Count / Population)
census_pd["Poverty Rate"] = 100 * \
    census_pd["Poverty Count"].astype(
        int) / census_pd["Population"].astype(int)

# Add in Employment Rate (Employment Count / Population)
census_pd["Unemployment Rate"] = 100 * \
    census_pd["Unemployment Count"].astype(
        int) / census_pd["Population"].astype(int)

# Final DataFrame
census_pd = census_pd[["County", "Name", "Population", "Median Age", "Household Income",
                       "Per Capita Income", "Poverty Count", "Poverty Rate", "Unemployment Rate"]]

census_pd.head()
```

	County	Name	Population	Median Age	Household Income	Per Capita Income	Poverty Count	Poverty Rate	Unemployment Rate
0	051	Fayette County, Illinois	21565.0	41.9	46650.0	23194.0	3421.0	15.863668	2.434500
1	107	Logan County, Illinois	29003.0	40.1	57308.0	27546.0	2323.0	8.009516	2.544564
2	165	Saline County, Illinois	23994.0	42.2	44090.0	25342.0	4936.0	20.571810	3.400850

Data Fetching and Cleaning (ETL)



```
# Importing data files

census_df = pd.read_csv("resources/census_us_county_outpu
education_df = pd.read_csv("resources/ers_usda_education.
county_df = pd.read_csv("resources/county_fips.csv")
```

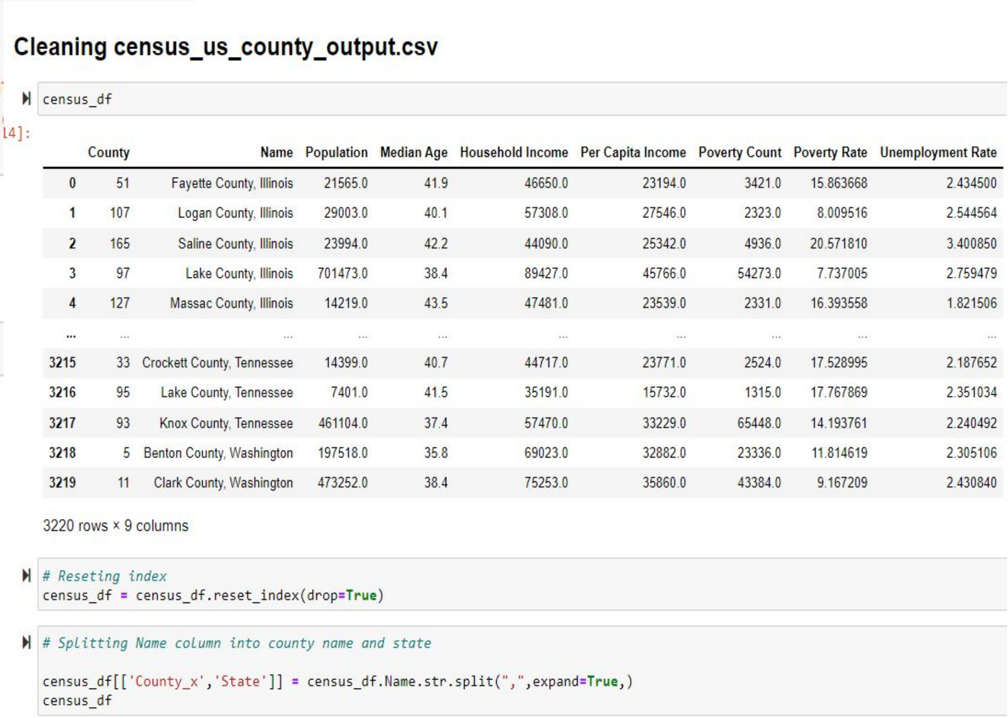


Cleaning County_fips.csv

```
county_df
```

[3]:

	Fips	County_Name	State_Abbr	State_Name
0	1001	Autauga County	AL	Alabama
1	1003	Baldwin County	AL	Alabama
2	1005	Barbour County	AL	Alabama
3	1007	Bibb County	AL	Alabama



Cleaning census_us_county_output.csv

```
census_df
```

14]:

	County	Name	Population	Median Age	Household Income	Per Capita Income	Poverty Count	Poverty Rate	Unemployment Rate
0	51	Fayette County, Illinois	21565.0	41.9	46650.0	23194.0	3421.0	15.863668	2.434500
1	107	Logan County, Illinois	29003.0	40.1	57308.0	27546.0	2323.0	8.009516	2.544564
2	165	Saline County, Illinois	23994.0	42.2	44090.0	25342.0	4936.0	20.571810	3.400850
3	97	Lake County, Illinois	701473.0	38.4	89427.0	45766.0	54273.0	7.737005	2.759479
4	127	Massac County, Illinois	14219.0	43.5	47481.0	23539.0	2331.0	16.393558	1.821506
...	...	...	...	...	...	...	...	...	...
3215	33	Crockett County, Tennessee	14399.0	40.7	44717.0	23771.0	2524.0	17.528995	2.187652
3216	95	Lake County, Tennessee	7401.0	41.5	35191.0	15732.0	1315.0	17.767869	2.351034
3217	93	Knox County, Tennessee	461104.0	37.4	57470.0	33229.0	65448.0	14.193761	2.240492
3218	5	Benton County, Washington	197518.0	35.8	69023.0	32882.0	23336.0	11.814619	2.305106
3219	11	Clark County, Washington	473252.0	38.4	75253.0	35860.0	43384.0	9.167209	2.430840

3220 rows × 9 columns

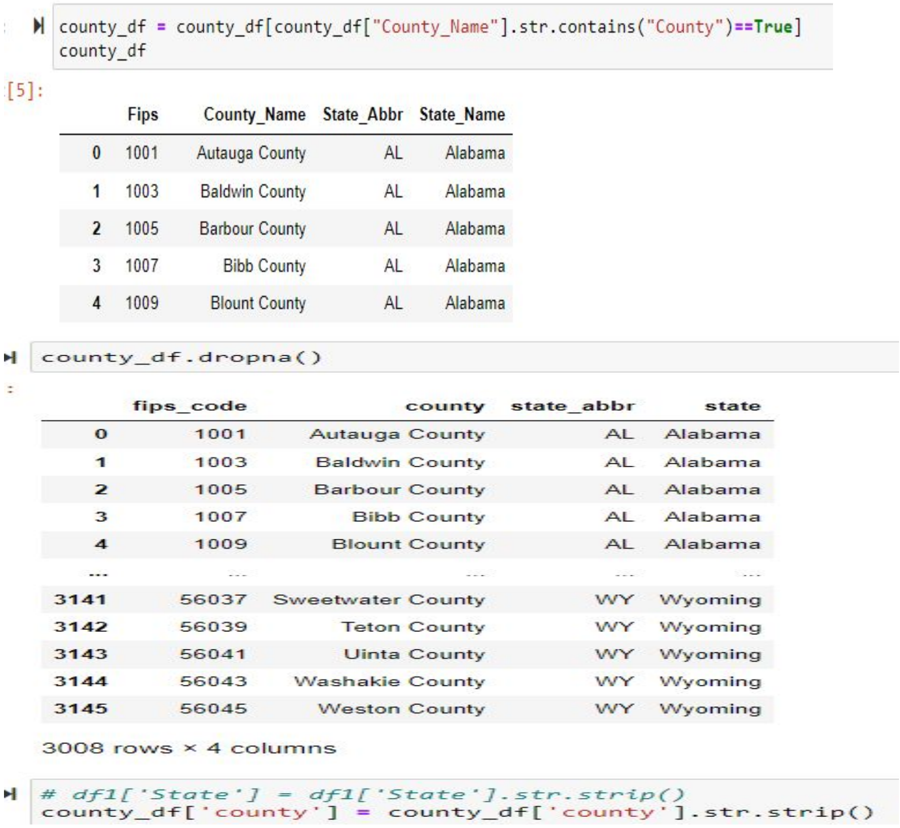
```
# Reseting index
census_df = census_df.reset_index(drop=True)
```

```
# Splitting Name column into county name and state

census_df[['County_x','State']] = census_df.Name.str.split(",",expand=True,)
census_df
```

Data Fetching and Cleaning (ETL)

Data is read into Pandas dataframe and following steps are performed for the cleaning process:

- Resetting the index.
- Renaming the columns.
- Dropping the NULLs rows and columns.
- Slicing extra strings in the column values like ‘county’.
- Stripping the blanks and commas.
- Splitting the column in the ‘census’ dataset.
- Dropping the duplicates.
- Removing the extra columns.
- Sorting the data frames based on state and county.
- Renaming the columns for preparing for merge.
- Merging and splitting of data frames is performed to have a primary key in each dataset to make it ready for the relational database.



```
county_df = county_df[county_df["County_Name"].str.contains("County")==True]
county_df
```

[5]:

	Fips	County_Name	State_Abbr	State_Name
0	1001	Autauga County	AL	Alabama
1	1003	Baldwin County	AL	Alabama
2	1005	Barbour County	AL	Alabama
3	1007	Bibb County	AL	Alabama
4	1009	Blount County	AL	Alabama

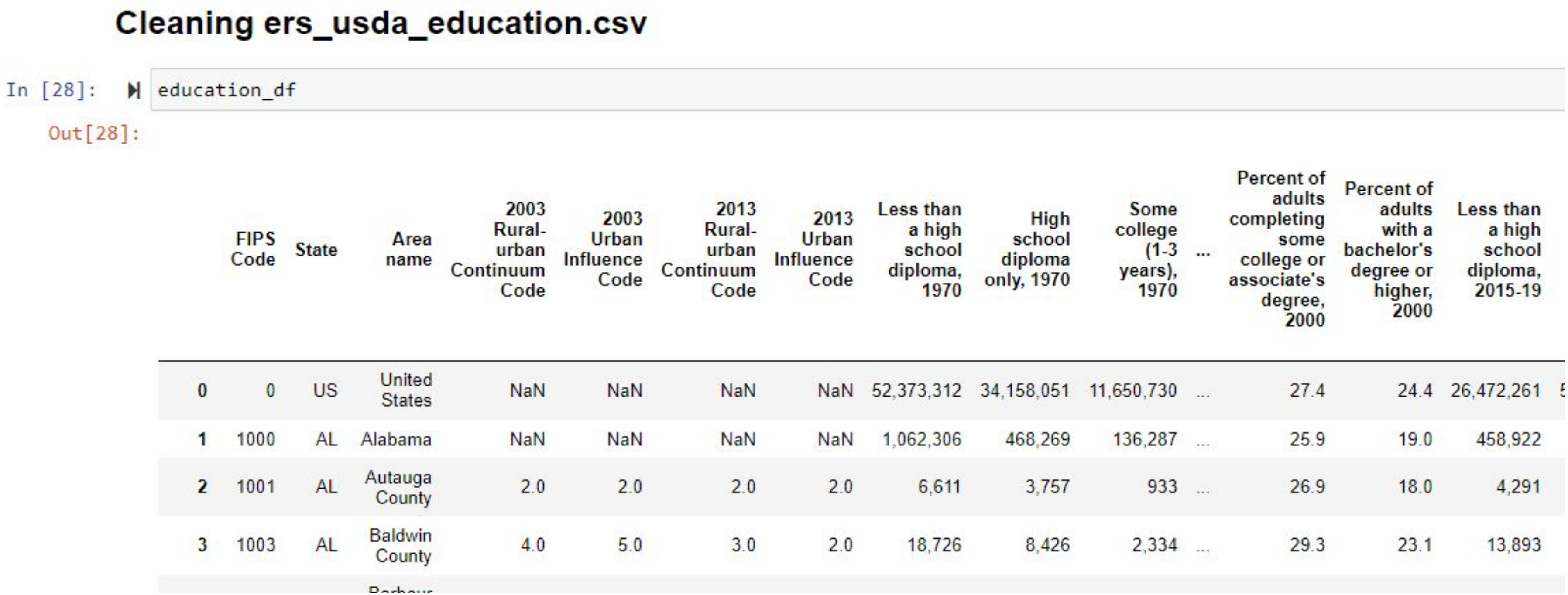
```
county_df.dropna()
```

	fips_code	county	state_abbr	state
0	1001	Autauga County	AL	Alabama
1	1003	Baldwin County	AL	Alabama
2	1005	Barbour County	AL	Alabama
3	1007	Bibb County	AL	Alabama
4	1009	Blount County	AL	Alabama
...	...	...	...	...
3141	56037	Sweetwater County	WY	Wyoming
3142	56039	Teton County	WY	Wyoming
3143	56041	Uinta County	WY	Wyoming
3144	56043	Washakie County	WY	Wyoming
3145	56045	Weston County	WY	Wyoming

3008 rows × 4 columns

```
# df1['State'] = df1['State'].str.strip()
county_df['county'] = county_df['county'].str.strip()
```

Data Fetching and Cleaning (ETL)



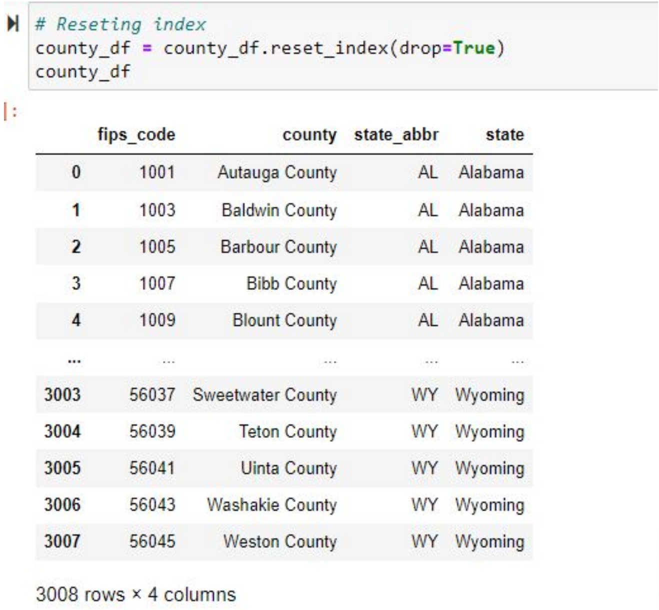
Cleaning ers_usda_education.csv

```
In [28]:  education_df
```

```
Out[28]:
```

	FIPS Code	State	Area name	2003 Rural-urban Continuum Code	2003 Urban Influence Code	2013 Rural-urban Continuum Code	2013 Urban Influence Code	Less than a high school diploma, 1970	High school diploma only, 1970	Some college (1-3 years), 1970	...	Percent of adults completing some college or associate's degree, 2000	Percent of adults with a bachelor's degree or higher, 2000	Less than a high school diploma, 2015-19
0	0	US	United States	NaN	NaN	NaN	NaN	52,373,312	34,158,051	11,650,730	...	27.4	24.4	26,472,261
1	1000	AL	Alabama	NaN	NaN	NaN	NaN	1,062,306	468,269	136,287	...	25.9	19.0	458,922
2	1001	AL	Autauga County	2.0	2.0	2.0	2.0	6,611	3,757	933	...	26.9	18.0	4,291
3	1003	AL	Baldwin County	4.0	5.0	3.0	2.0	18,726	8,426	2,334	...	29.3	23.1	13,893
			Barbour											

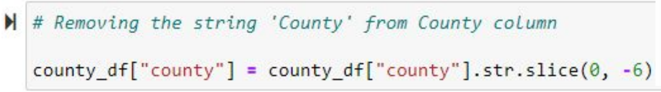
Data Fetching and Cleaning (ETL)



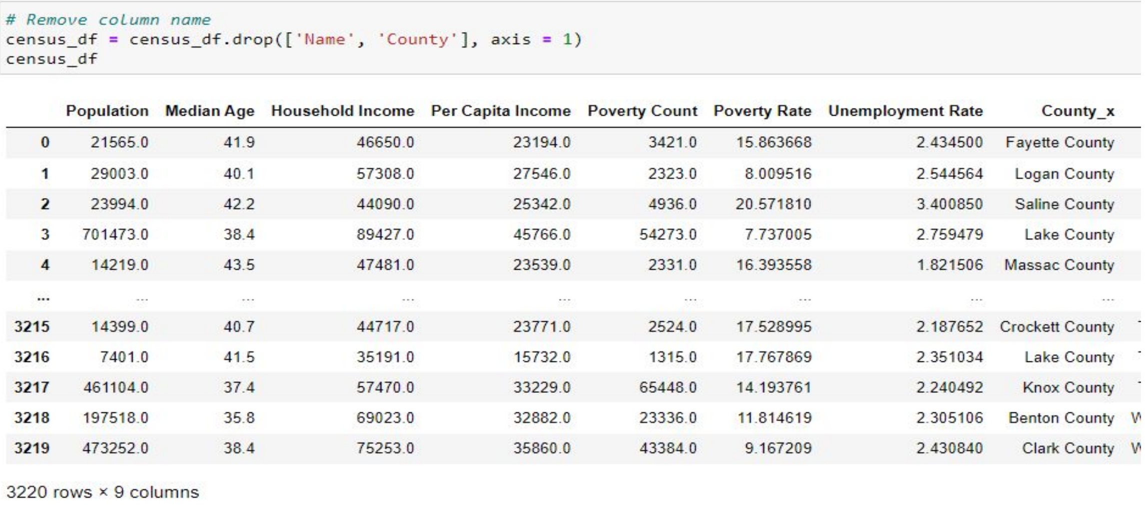
```
# Reseting index
county_df = county_df.reset_index(drop=True)
county_df
```

	fips_code	county	state_abbr	state
0	1001	Autauga County	AL	Alabama
1	1003	Baldwin County	AL	Alabama
2	1005	Barbour County	AL	Alabama
3	1007	Bibb County	AL	Alabama
4	1009	Blount County	AL	Alabama
...	...	...	...	...
3003	56037	Sweetwater County	WY	Wyoming
3004	56039	Teton County	WY	Wyoming
3005	56041	Uinta County	WY	Wyoming
3006	56043	Washakie County	WY	Wyoming
3007	56045	Weston County	WY	Wyoming

3008 rows × 4 columns



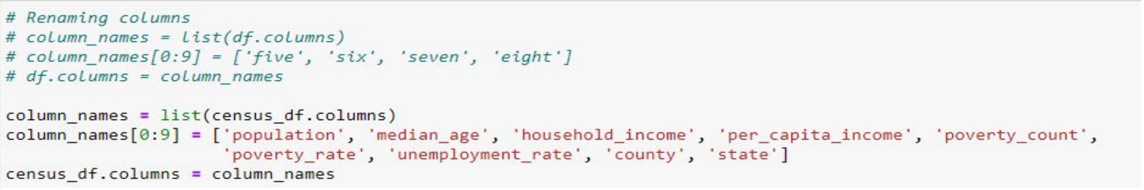
```
# Removing the string 'County' from County column

county_df["county"] = county_df["county"].str.slice(0, -6)
```



```
# Remove column name
census_df = census_df.drop(['Name', 'County'], axis = 1)
census_df
```

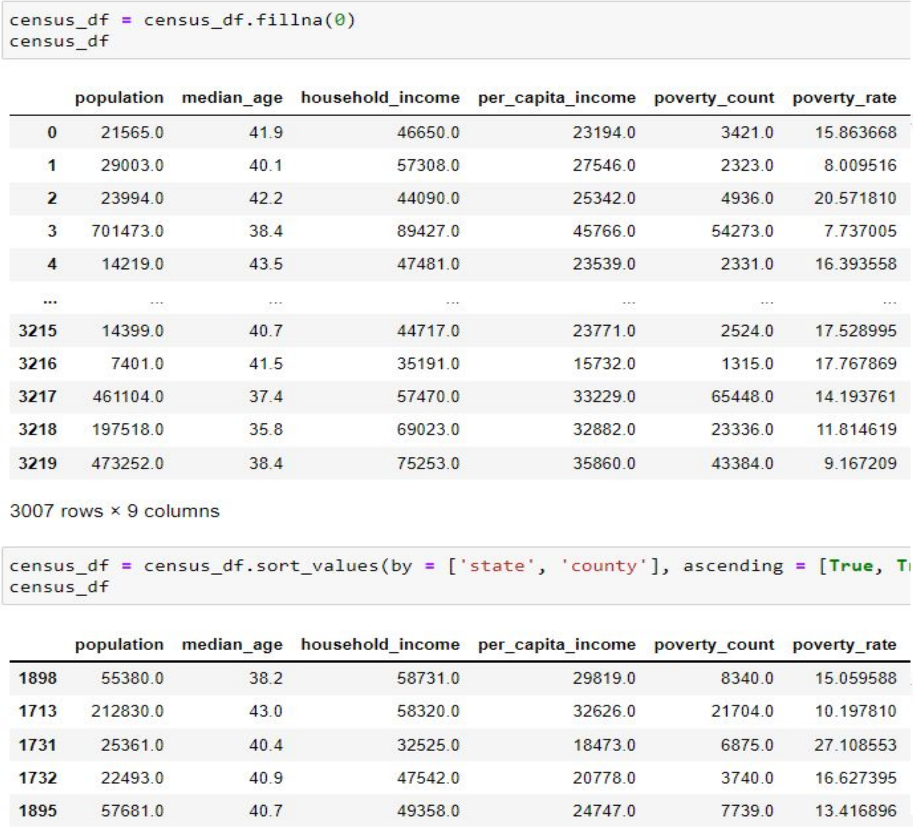
	Population	Median Age	Household Income	Per Capita Income	Poverty Count	Poverty Rate	Unemployment Rate	County_x	
0	21565.0	41.9	46650.0	23194.0	3421.0	15.863668	2.434500	Fayette County	
1	29003.0	40.1	57308.0	27546.0	2323.0	8.009516	2.544564	Logan County	
2	23994.0	42.2	44090.0	25342.0	4936.0	20.571810	3.400850	Saline County	
3	701473.0	38.4	89427.0	45766.0	54273.0	7.737005	2.759479	Lake County	
4	14219.0	43.5	47481.0	23539.0	2331.0	16.393558	1.821506	Massac County	
...	...	...	...	...	...	...	...	...	
3215	14399.0	40.7	44717.0	23771.0	2524.0	17.528995	2.187652	Crockett County	T
3216	7401.0	41.5	35191.0	15732.0	1315.0	17.767869	2.351034	Lake County	T
3217	461104.0	37.4	57470.0	33229.0	65448.0	14.193761	2.240492	Knox County	T
3218	197518.0	35.8	69023.0	32882.0	23336.0	11.814619	2.305106	Benton County	W
3219	473252.0	38.4	75253.0	35860.0	43384.0	9.167209	2.430840	Clark County	W

3220 rows × 9 columns



```
# Renaming columns
# column_names = list(df.columns)
# column_names[0:9] = ['five', 'six', 'seven', 'eight']
# df.columns = column_names

column_names = list(census_df.columns)
column_names[0:9] = ['population', 'median_age', 'household_income', 'per_capita_income', 'poverty_count',
                     'poverty_rate', 'unemployment_rate', 'county', 'state']
census_df.columns = column_names
```

Data Fetching and Cleaning (ETL)



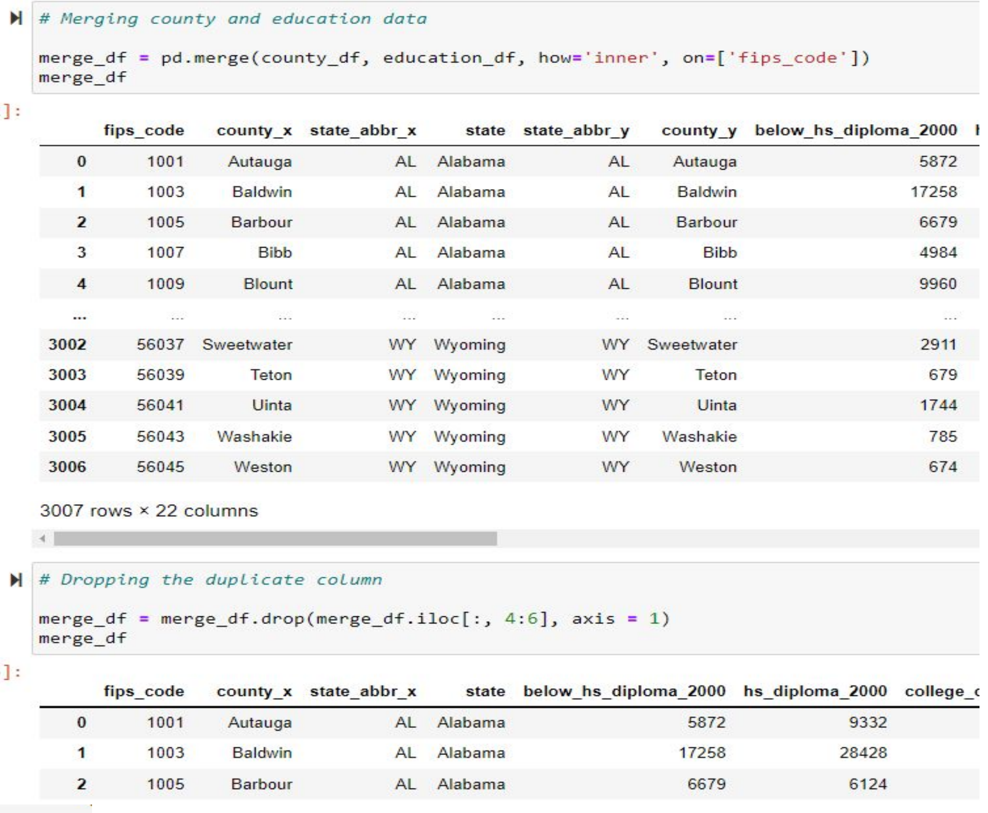
```
census_df = census_df.fillna(0)
census_df
```

	population	median_age	household_income	per_capita_income	poverty_count	poverty_rate
0	21565.0	41.9	46650.0	23194.0	3421.0	15.863668
1	29003.0	40.1	57308.0	27546.0	2323.0	8.009516
2	23994.0	42.2	44090.0	25342.0	4936.0	20.571810
3	701473.0	38.4	89427.0	45766.0	54273.0	7.737005
4	14219.0	43.5	47481.0	23539.0	2331.0	16.393558
...	...	...	...	...	...	...
3215	14399.0	40.7	44717.0	23771.0	2524.0	17.528995
3216	7401.0	41.5	35191.0	15732.0	1315.0	17.767869
3217	461104.0	37.4	57470.0	33229.0	65448.0	14.193761
3218	197518.0	35.8	69023.0	32882.0	23336.0	11.814619
3219	473252.0	38.4	75253.0	35860.0	43384.0	9.167209

3007 rows × 9 columns

```
census_df = census_df.sort_values(by = ['state', 'county'], ascending = [True, Tr
census_df
```

	population	median_age	household_income	per_capita_income	poverty_count	poverty_rate
1898	55380.0	38.2	58731.0	29819.0	8340.0	15.059588
1713	212830.0	43.0	58320.0	32626.0	21704.0	10.197810
1731	25361.0	40.4	32525.0	18473.0	6875.0	27.108553
1732	22493.0	40.9	47542.0	20778.0	3740.0	16.627395
1895	57681.0	40.7	49358.0	24747.0	7739.0	13.416896



```
# Merging county and education data

merge_df = pd.merge(county_df, education_df, how='inner', on=['fips_code'])
merge_df
```

2]:

	fips_code	county_x	state_abbr_x	state	state_abbr_y	county_y	below_hs_diploma_2000
0	1001	Autauga	AL	Alabama	AL	Autauga	5872
1	1003	Baldwin	AL	Alabama	AL	Baldwin	17258
2	1005	Barbour	AL	Alabama	AL	Barbour	6679
3	1007	Bibb	AL	Alabama	AL	Bibb	4984
4	1009	Blount	AL	Alabama	AL	Blount	9960
...	...	...	...	...	...	...	...
3002	56037	Sweetwater	WY	Wyoming	WY	Sweetwater	2911
3003	56039	Teton	WY	Wyoming	WY	Teton	679
3004	56041	Uinta	WY	Wyoming	WY	Uinta	1744
3005	56043	Washakie	WY	Wyoming	WY	Washakie	785
3006	56045	Weston	WY	Wyoming	WY	Weston	674

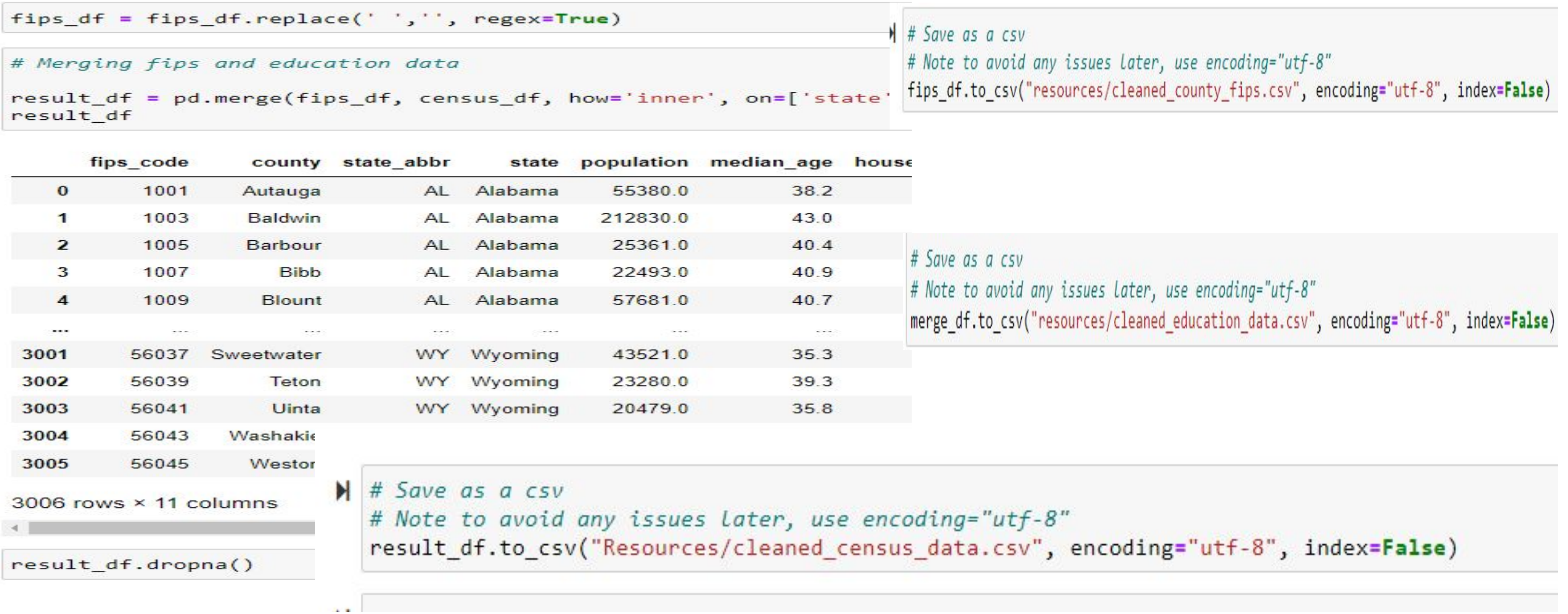
3007 rows × 22 columns

```
# Dropping the duplicate column

merge_df = merge_df.drop(merge_df.iloc[:, 4:6], axis = 1)
merge_df
```

3]:

	fips_code	county_x	state_abbr_x	state	below_hs_diploma_2000	hs_diploma_2000
0	1001	Autauga	AL	Alabama	5872	9332
1	1003	Baldwin	AL	Alabama	17258	28428
2	1005	Barbour	AL	Alabama	6679	6124

Data Fetching and Cleaning (ETL)



```
fips_df = fips_df.replace(' ','', regex=True)
```

```
# Merging fips and education data

result_df = pd.merge(fips_df, census_df, how='inner', on=['state'
result_df
```

```
      fips_code     county  state_abbr    state  population  median_age  house
   0       1001    Autauga          AL  Alabama     55380.0        38.2
   1       1003    Baldwin          AL  Alabama    212830.0        43.0
   2       1005    Barbour          AL  Alabama     25361.0        40.4
   3       1007       Bibb          AL  Alabama     22493.0        40.9
   4       1009     Blount          AL  Alabama     57681.0        40.7
 ...        ...        ...         ...      ...         ...         ...
3001      56037 Sweetwater          WY  Wyoming     43521.0        35.3
3002      56039      Teton          WY  Wyoming     23280.0        39.3
3003      56041      Uinta          WY  Wyoming     20479.0        35.8
3004      56043   Washakie
3005      56045     Westor

3006 rows × 11 columns
```

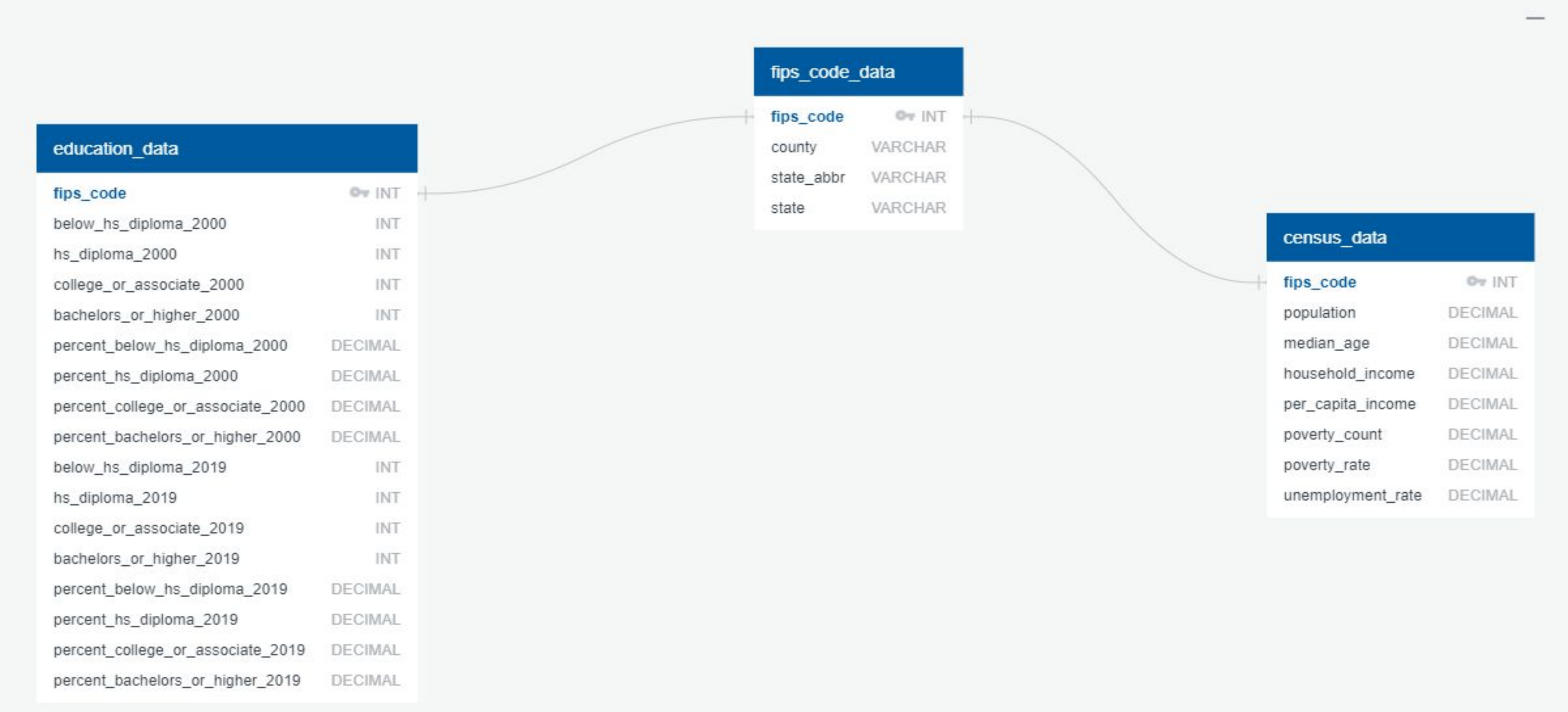
```
result_df.dropna()
```

```
# Save as a csv
# Note to avoid any issues later, use encoding="utf-8"
fips_df.to_csv("resources/cleaned_county_fips.csv", encoding="utf-8", index=False)
```

```
# Save as a csv
# Note to avoid any issues later, use encoding="utf-8"
merge_df.to_csv("resources/cleaned_education_data.csv", encoding="utf-8", index=False)
```

```
# Save as a csv
# Note to avoid any issues later, use encoding="utf-8"
result_df.to_csv("Resources/cleaned_census_data.csv", encoding="utf-8", index=False)
```

Designing the Relational Database For PostgreSQL



education_data	
fips_code	INT
below_hs_diploma_2000	INT
hs_diploma_2000	INT
college_or_associate_2000	INT
bachelors_or_higher_2000	INT
percent_below_hs_diploma_2000	DECIMAL
percent_hs_diploma_2000	DECIMAL
percent_college_or_associate_2000	DECIMAL
percent_bachelors_or_higher_2000	DECIMAL
below_hs_diploma_2019	INT
hs_diploma_2019	INT
college_or_associate_2019	INT
bachelors_or_higher_2019	INT
percent_below_hs_diploma_2019	DECIMAL
percent_hs_diploma_2019	DECIMAL
percent_college_or_associate_2019	DECIMAL
percent_bachelors_or_higher_2019	DECIMAL

fips_code_data	
fips_code	INT
county	VARCHAR
state_abbr	VARCHAR
state	VARCHAR

census_data	
fips_code	INT
population	DECIMAL
median_age	DECIMAL
household_income	DECIMAL
per_capita_income	DECIMAL
poverty_count	DECIMAL
poverty_rate	DECIMAL
unemployment_rate	DECIMAL

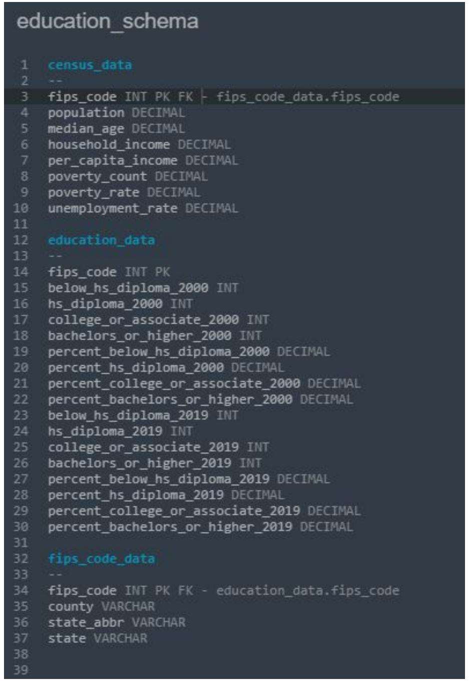
Designing the Relational Database For PostgreSQL

Relational database design diagram is drawn using an online tool ‘Quick DBD’.

Primary and Foreign keys are assigned and tables are created.

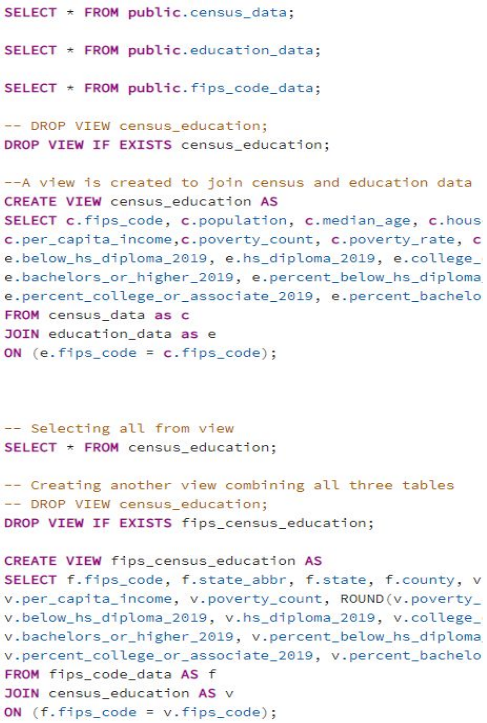
Finally, datasets are imported in PostgeSQL database.

Queries are performed for testing and views are created here.



education_schema

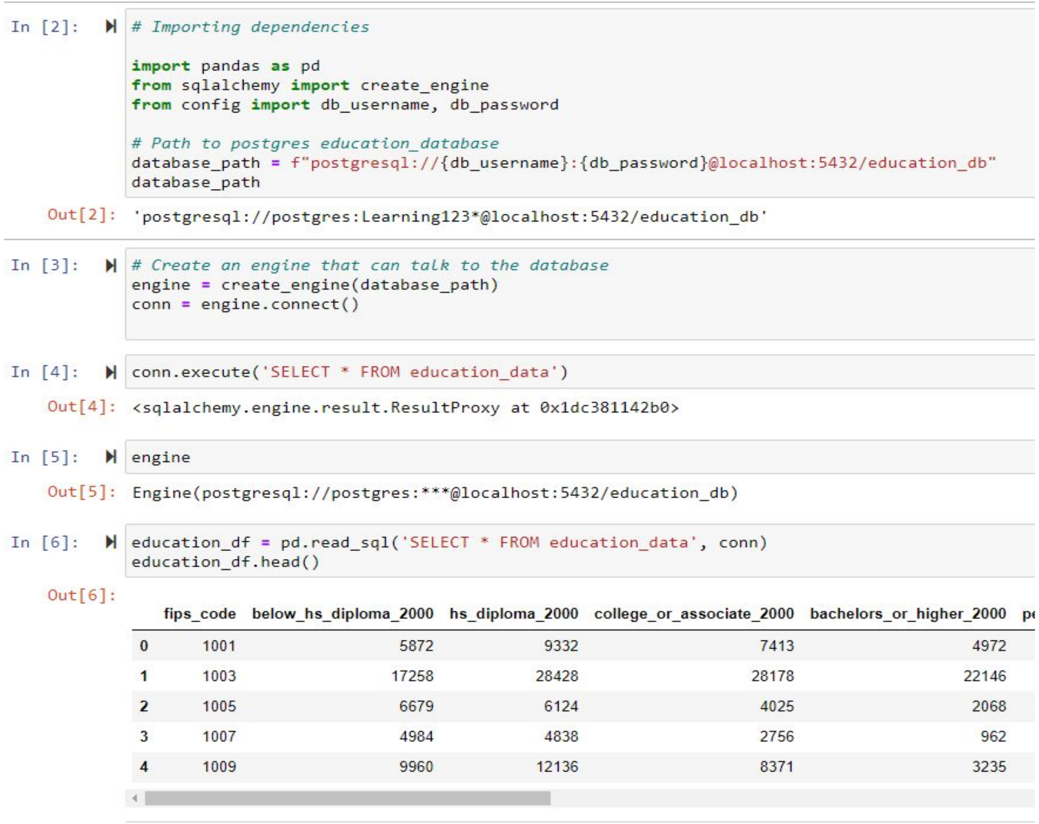
```
census_data
--
fips_code INT PK FK - fips_code_data.fips_code
population DECIMAL
median_age DECIMAL
household_income DECIMAL
per_capita_income DECIMAL
poverty_count DECIMAL
poverty_rate DECIMAL
unemployment_rate DECIMAL

education_data
--
fips_code INT PK
below_hs_diploma_2000 INT
hs_diploma_2000 INT
college_or_associate_2000 INT
bachelors_or_higher_2000 INT
percent_below_hs_diploma_2000 DECIMAL
percent_hs_diploma_2000 DECIMAL
percent_college_or_associate_2000 DECIMAL
percent_bachelors_or_higher_2000 DECIMAL
below_hs_diploma_2019 INT
hs_diploma_2019 INT
college_or_associate_2019 INT
bachelors_or_higher_2019 INT
percent_below_hs_diploma_2019 DECIMAL
percent_hs_diploma_2019 DECIMAL
percent_college_or_associate_2019 DECIMAL
percent_bachelors_or_higher_2019 DECIMAL

fips_code_data
--
fips_code INT PK FK - education_data.fips_code
county VARCHAR
state_abbr VARCHAR
state VARCHAR
```



```
SELECT * FROM public.census_data;

SELECT * FROM public.education_data;

SELECT * FROM public.fips_code_data;

-- DROP VIEW census_education;
DROP VIEW IF EXISTS census_education;

--A view is created to join census and education data
CREATE VIEW census_education AS
SELECT c.fips_code, c.population, c.median_age, c.hous
c.per_capita_income,c.poverty_count, c.poverty_rate, c
e.below_hs_diploma_2019, e.hs_diploma_2019, e.college_
e.bachelors_or_higher_2019, e.percent_below_hs_diploma
e.percent_college_or_associate_2019, e.percent_bachelo
FROM census_data as c
JOIN education_data as e
ON (e.fips_code = c.fips_code);



-- Selecting all from view
SELECT * FROM census_education;

-- Creating another view combining all three tables
-- DROP VIEW census_education;
DROP VIEW IF EXISTS fips_census_education;

CREATE VIEW fips_census_education AS
SELECT f.fips_code, f.state_abbr, f.state, f.county, v
v.per_capita_income, v.poverty_count, ROUND(v.poverty_
v.below_hs_diploma_2019, v.hs_diploma_2019, v.college_
v.bachelors_or_higher_2019, v.percent_below_hs_diploma
v.percent_college_or_associate_2019, v.percent_bachelo
FROM fips_code_data AS f
JOIN census_education AS v
ON (f.fips_code = v.fips_code);
```

SQLAlchemy - The Python SQL Toolkit

The SQLAlchemy analysis is performed by importing the dependencies and connecting to the database engine. This allows the access of the database in Python environment (Jupyter notebook).



```
In [2]:  # Importing dependencies

         import pandas as pd
         from sqlalchemy import create_engine
         from config import db_username, db_password

         # Path to postgres education_database
         database_path = f"postgresql://{db_username}:{db_password}@localhost:5432/education_db"
         database_path

Out[2]:  'postgresql://postgres:Learning123*@localhost:5432/education_db'
```

```
In [3]:  # Create an engine that can talk to the database
         engine = create_engine(database_path)
         conn = engine.connect()
```

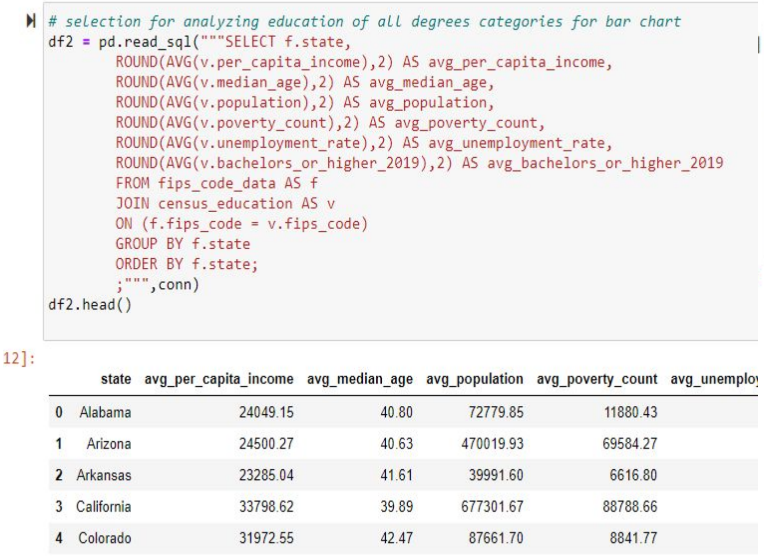
```
In [4]:  conn.execute('SELECT * FROM education_data')

Out[4]:  <sqlalchemy.engine.result.ResultProxy at 0x1dc381142b0>
```

```
In [5]:  engine

Out[5]:  Engine(postgresql://postgres:***@localhost:5432/education_db)
```

```
In [6]:  education_df = pd.read_sql('SELECT * FROM education_data', conn)
         education_df.head()

Out[6]:
```

	fips_code	below_hs_diploma_2000	hs_diploma_2000	college_or_associate_2000	bachelors_or_higher_2000	pe
0	1001	5872	9332	7413	4972	
1	1003	17258	28428	28178	22146	
2	1005	6679	6124	4025	2068	
3	1007	4984	4838	2756	962	
4	1009	9960	12136	8371	3235	

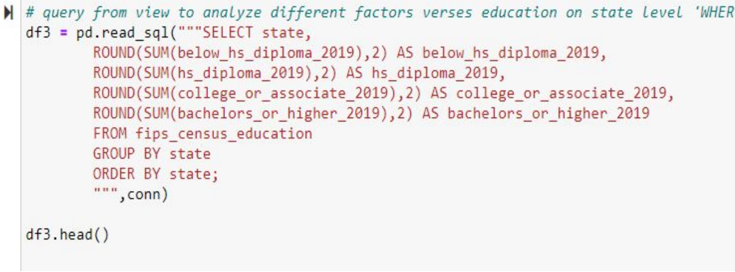
SQLAlchemy - The Python SQL Toolkit



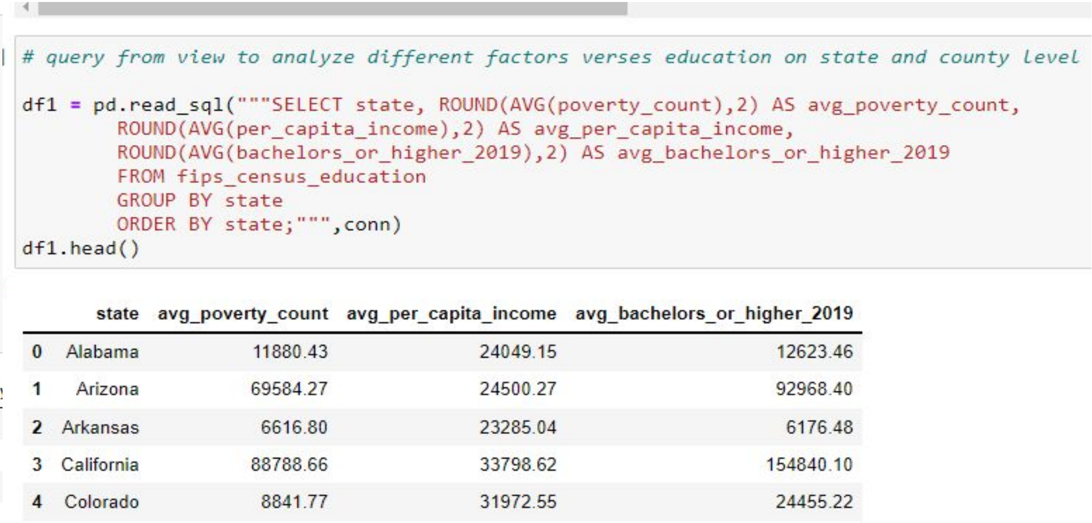
```
# selection for analyzing education of all degrees categories for bar chart
df2 = pd.read_sql("""SELECT f.state,
        ROUND(AVG(v.per_capita_income),2) AS avg_per_capita_income,
        ROUND(AVG(v.median_age),2) AS avg_median_age,
        ROUND(AVG(v.population),2) AS avg_population,
        ROUND(AVG(v.poverty_count),2) AS avg_poverty_count,
        ROUND(AVG(v.unemployment_rate),2) AS avg_unemployment_rate,
        ROUND(AVG(v.bachelors_or_higher_2019),2) AS avg_bachelors_or_higher_2019
        FROM fips_code_data AS f
        JOIN census_education AS v
        ON (f.fips_code = v.fips_code)
        GROUP BY f.state
        ORDER BY f.state;
        ;""",conn)
df2.head()
```

12]:

	state	avg_per_capita_income	avg_median_age	avg_population	avg_poverty_count	avg_unemploy
0	Alabama	24049.15	40.80	72779.85	11880.43	
1	Arizona	24500.27	40.63	470019.93	69584.27	
2	Arkansas	23285.04	41.61	39991.60	6616.80	
3	California	33798.62	39.89	677301.67	88788.66	
4	Colorado	31972.55	42.47	87661.70	8841.77	

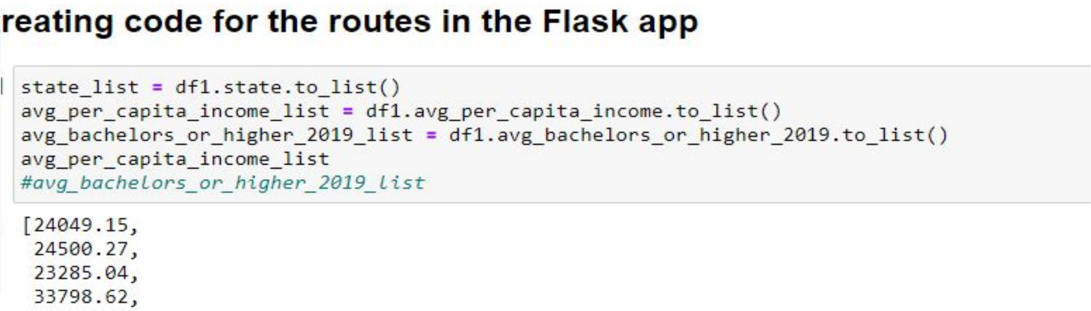


```
# query from view to analyze different factors verses education on state level 'WHER
df3 = pd.read_sql("""SELECT state,
        ROUND(SUM(below_hs_diploma_2019),2) AS below_hs_diploma_2019,
        ROUND(SUM(hs_diploma_2019),2) AS hs_diploma_2019,
        ROUND(SUM(college_or_associate_2019),2) AS college_or_associate_2019,
        ROUND(SUM(bachelors_or_higher_2019),2) AS bachelors_or_higher_2019
        FROM fips_census_education
        GROUP BY state
        ORDER BY state;
        """,conn)

df3.head()
```



```
# query from view to analyze different factors verses education on state and county level

df1 = pd.read_sql("""SELECT state, ROUND(AVG(poverty_count),2) AS avg_poverty_count,
        ROUND(AVG(per_capita_income),2) AS avg_per_capita_income,
        ROUND(AVG(bachelors_or_higher_2019),2) AS avg_bachelors_or_higher_2019
        FROM fips_census_education
        GROUP BY state
        ORDER BY state;""",conn)
df1.head()
```

	state	avg_poverty_count	avg_per_capita_income	avg_bachelors_or_higher_2019
0	Alabama	11880.43	24049.15	12623.46
1	Arizona	69584.27	24500.27	92968.40
2	Arkansas	6616.80	23285.04	6176.48
3	California	88788.66	33798.62	154840.10
4	Colorado	8841.77	31972.55	24455.22



reating code for the routes in the Flask app

```
state_list = df1.state.to_list()
avg_per_capita_income_list = df1.avg_per_capita_income.to_list()
avg_bachelors_or_higher_2019_list = df1.avg_bachelors_or_higher_2019.to_list()
avg_per_capita_income_list
#avg_bachelors_or_higher_2019_list
```

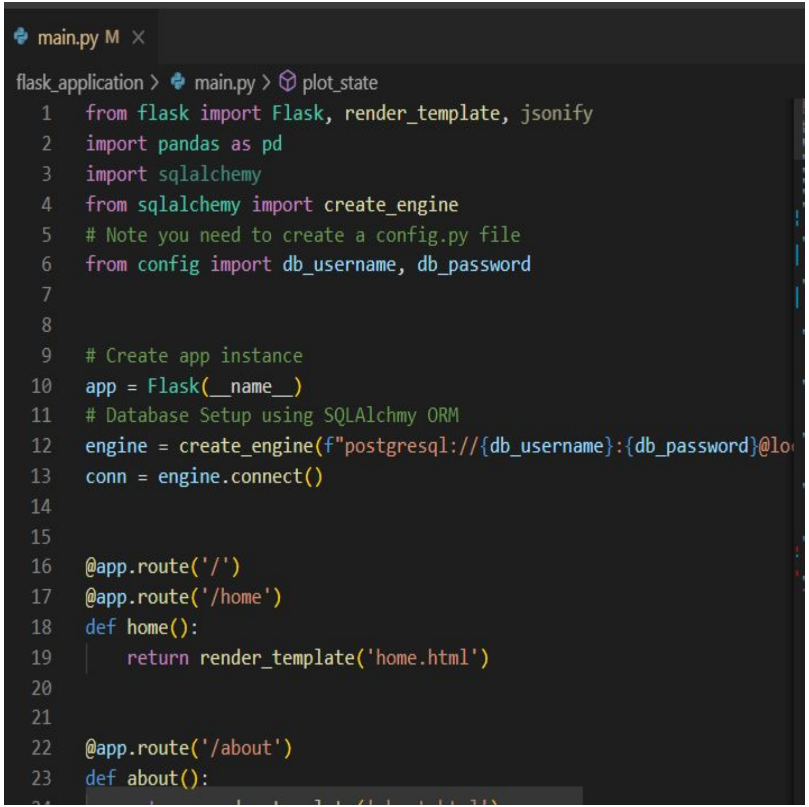
```
[24049.15,
 24500.27,
 23285.04,
 33798.62,
```

Flask Application, Javascript, HTML & CSS

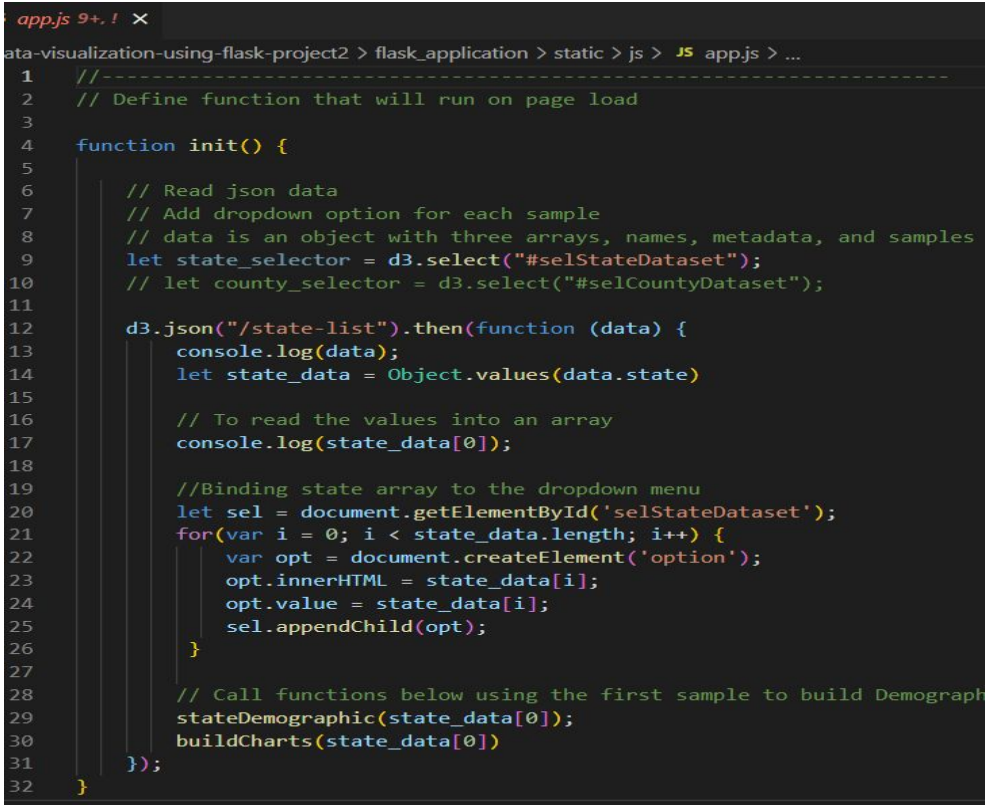
A dashboard is designed using the following files:

- main.py -- dependencies are imported, routes are created, and queries are added.
- app.js -- using d3.js for processing the data, and plotly.js and chart.js libraries are used for plotting charts.
- Index.html -- source links are added, dashboard is designed using these html files.
- about.html
- base.html
- home.html
- plotly.html
- leaflet.html

Flask Application, Javascript, HTML & CSS



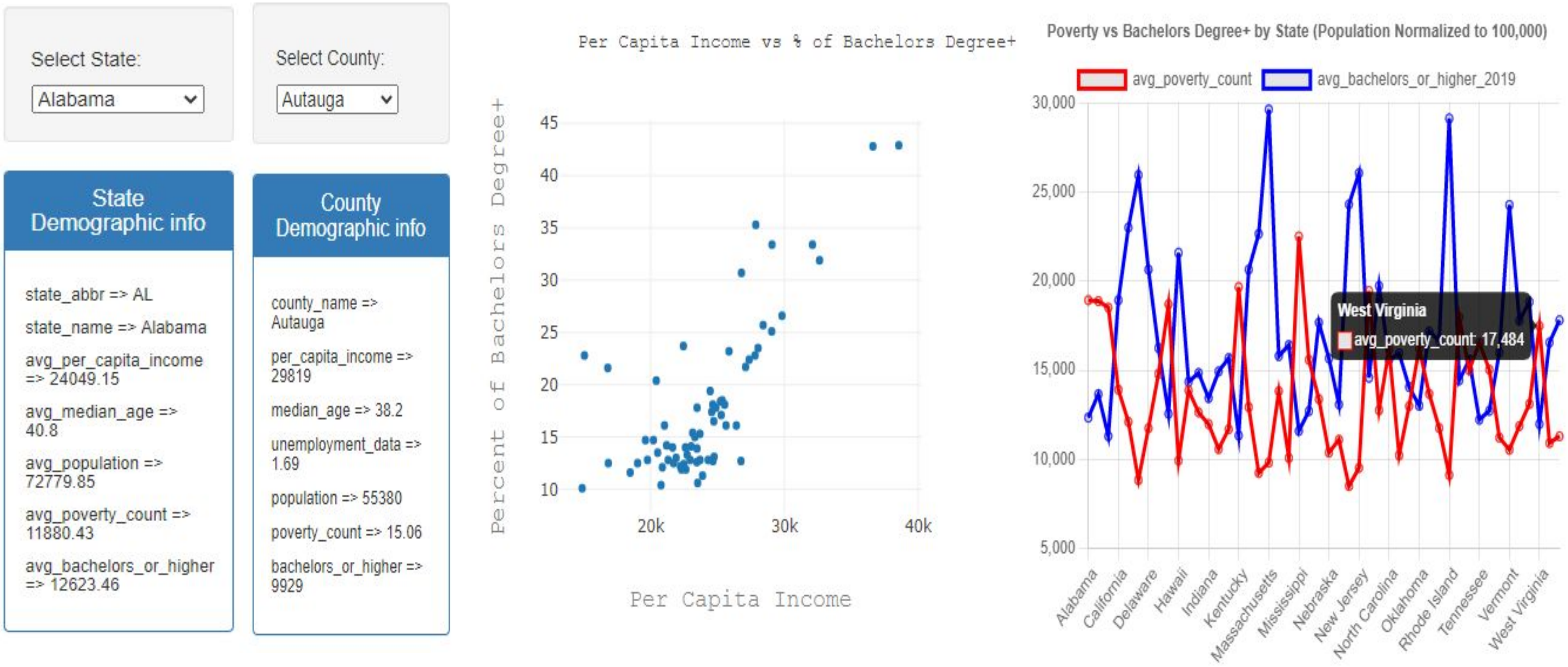
```python
from flask import Flask, render_template, jsonify
import pandas as pd
import sqlalchemy
from sqlalchemy import create_engine
# Note you need to create a config.py file
from config import db_username, db_password


# Create app instance
app = Flask(__name__)
# Database Setup using SQLAlchmy ORM
engine = create_engine(f"postgresql://{db_username}:{db_password}@lo
conn = engine.connect()


@app.route('/')
@app.route('/home')
def home():
    return render_template('home.html')


@app.route('/about')
def about():
```



app.js 9+, !
ata-visualization-using-flask-project2 > flask_application > static > js > JS app.js > ...

```javascript
//-----------------------------------------------------------------
// Define function that will run on page load

function init() {

    // Read json data
    // Add dropdown option for each sample
    // data is an object with three arrays, names, metadata, and samples
    let state_selector = d3.select("#selStateDataset");
    // let county_selector = d3.select("#selCountyDataset");

    d3.json("/state-list").then(function (data) {
        console.log(data);
        let state_data = Object.values(data.state)

        // To read the values into an array
        console.log(state_data[0]);

        //Binding state array to the dropdown menu
        let sel = document.getElementById('selStateDataset');
        for(var i = 0; i < state_data.length; i++) {
            var opt = document.createElement('option');
            opt.innerHTML = state_data[i];
            opt.value = state_data[i];
            sel.appendChild(opt);
        }

        // Call functions below using the first sample to build Demograph
        stateDemographic(state_data[0]);
        buildCharts(state_data[0])
    });
}
```

Dashboard Presentation



Select State:

Alabama

Select County:

Autauga

State Demographic info

state_abbr => AL

state_name => Alabama

avg_per_capita_income => 24049.15

avg_median_age => 40.8

avg_population => 72779.85

avg_poverty_count => 11880.43

avg_bachelors_or_higher => 12623.46

County Demographic info

county_name => Autauga

per_capita_income => 29819

median_age => 38.2

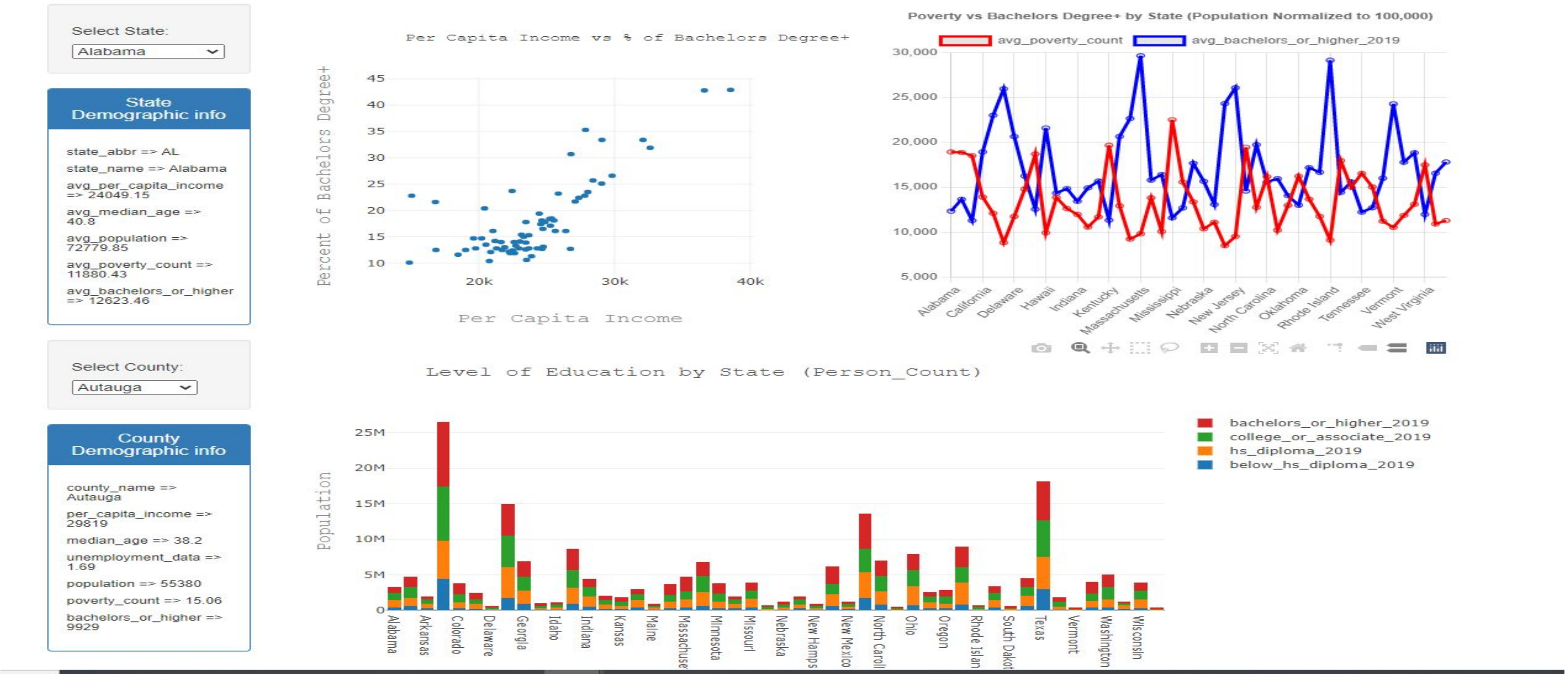
unemployment_data => 1.69

population => 55380

poverty_count => 15.06

bachelors_or_higher => 9929

Dashboard Presentation



Select State:

Alabama

State Demographic info

state_abbr => AL

state_name => Alabama

avg_per_capita_income => 24049.15

avg_median_age => 40.8

avg_population => 72779.85

avg_poverty_count => 11880.43

avg_bachelors_or_higher => 12623.46

Select County:

Autauga

County Demographic info

county_name => Autauga

per_capita_income => 29819

median_age => 38.2

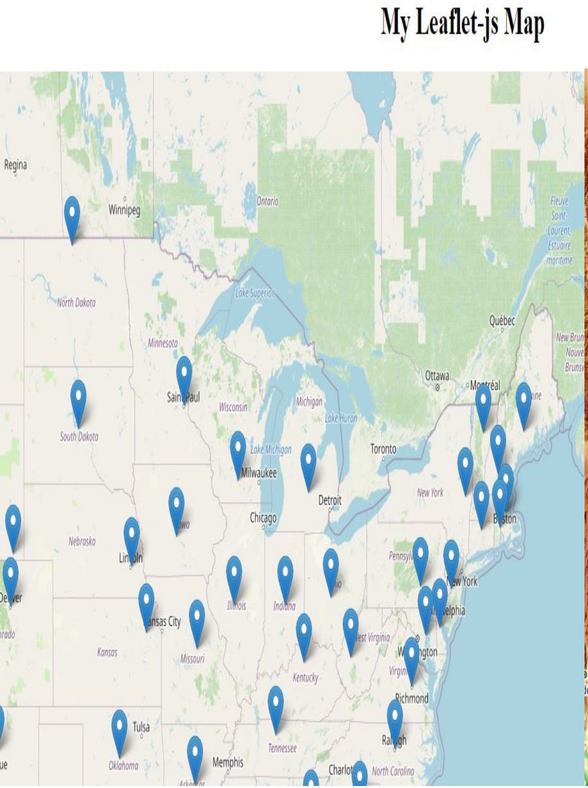
unemployment_data => 1.69

population => 55380

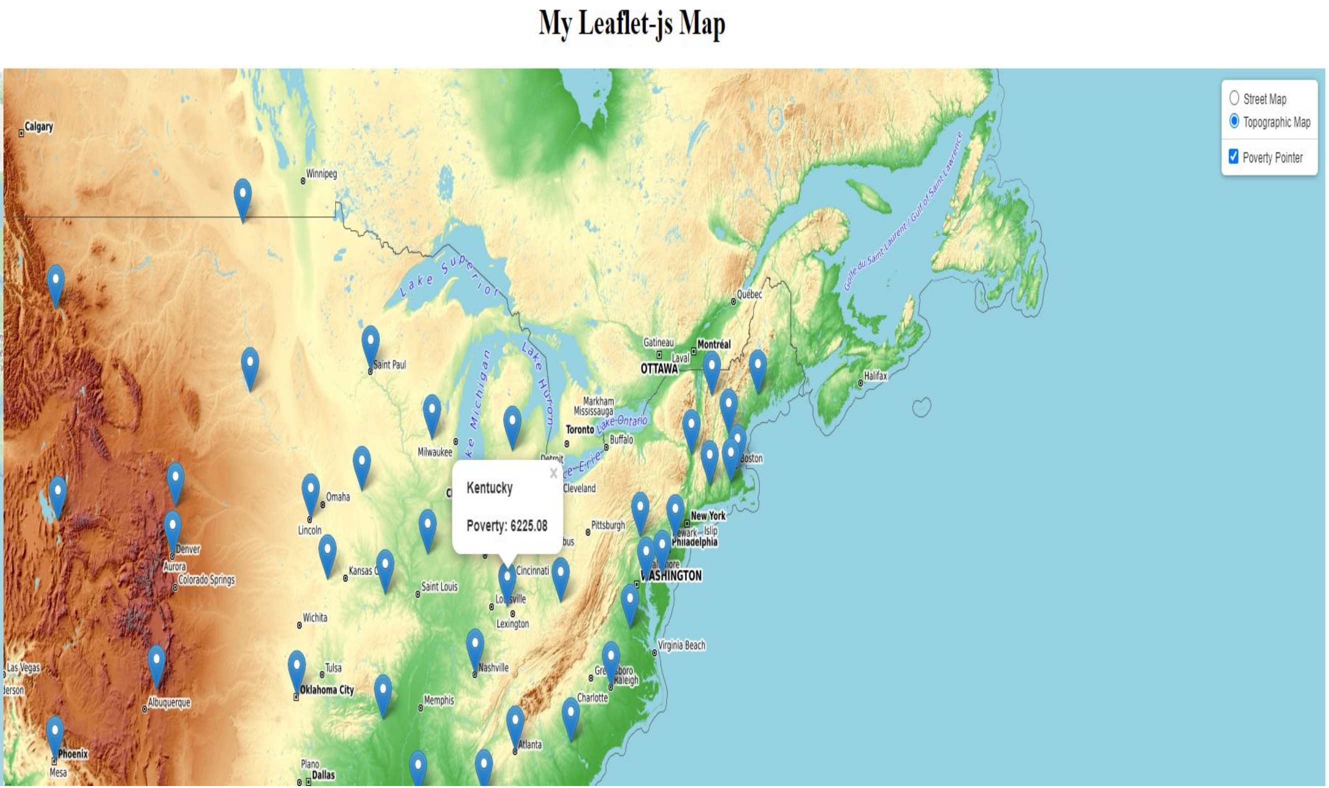
poverty_count => 15.06

bachelors_or_higher => 9929

Leaflet Map Page



My Leaflet-js Map



My Leaflet-js Map

○ Street Map
● Topographic Map

☑ Poverty Pointer

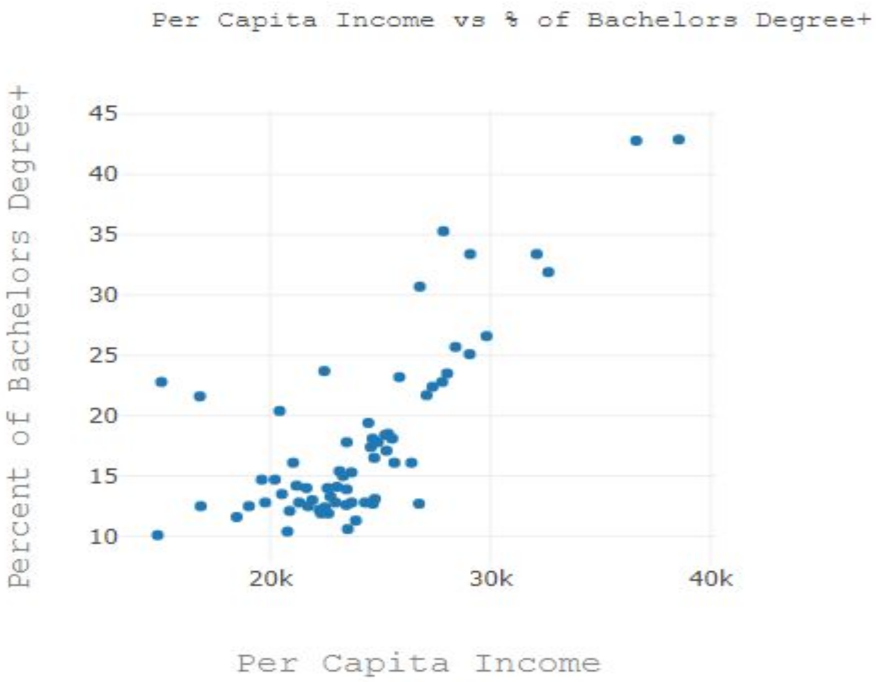
Kentucky ×

Poverty: 6225.08

Storytelling through Data

Scatter Plot:

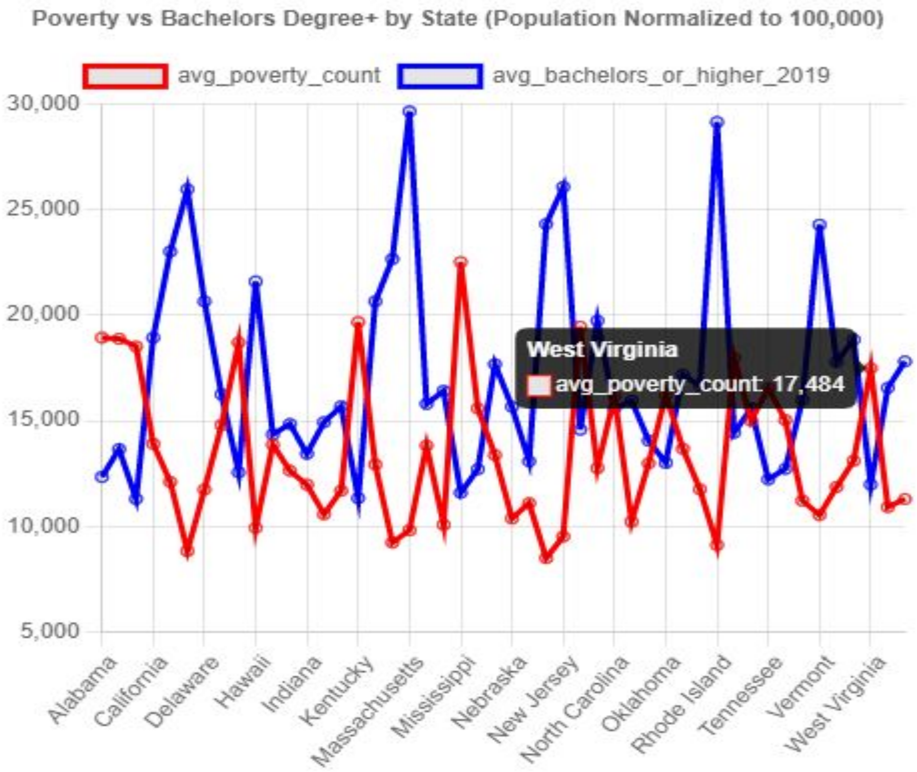
- This chart is responsive, representing counties of single state each time.
- There is a direct correlation between the per capita Income and the percentage of bachelor's degree And higher.
- As the per capita income goes up, so does the percentage of bachelor's degree and higher.



Storytelling through Data

Line Plot:

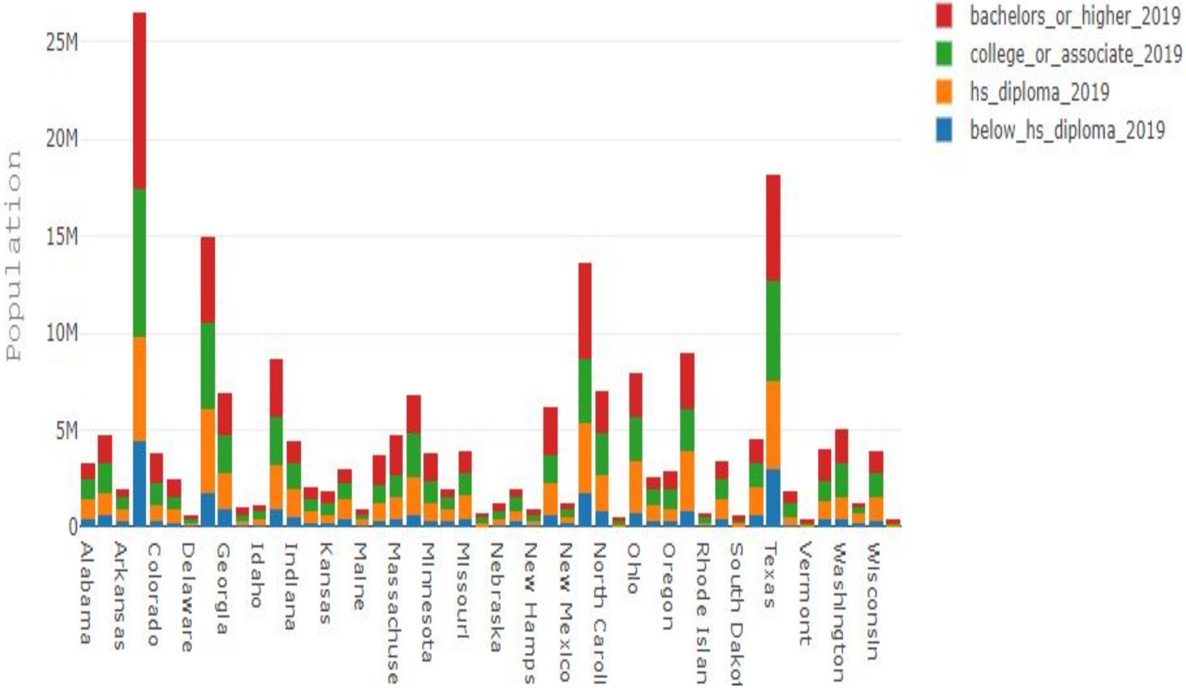
- Here every state’s population, poverty count and education is normalized by dividing the values with 100,000, so that we can compare one state to the other.
- There is a inverse relationship between the Poverty count and average bachelor’s and Higher degree, meaning, higher the poverty count, lower the education achievement.
- The line plot is not at the county level, Because there is not enough data to show the inverse relationship.



Storytelling through Data

Bar Plot:

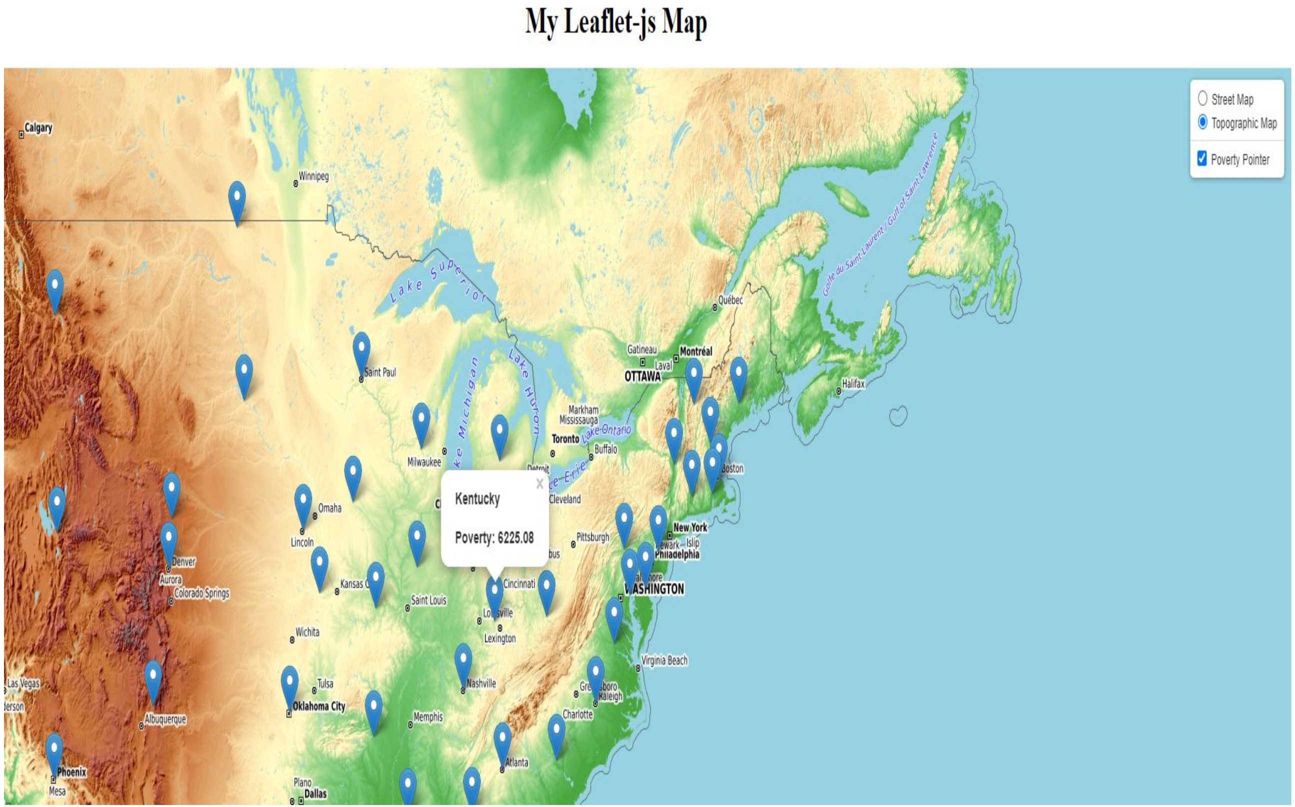
- This chart is for comparison Of the four categories of Education between the states.
- I choose to show the total of each degree, where when hover over, can see the difference in numbers.
- Wider the red portion, means higher the level of achievement.



Storytelling through Data

Leaflet Map:

It’s the visual representation of the poverty level in each state.



My Leaflet-js Map

○ Street Map
● Topographic Map
☑ Poverty Pointer

Kentucky

Poverty: 6225.08

Thank you
QuotesBlog.net