Data Collection and ETL Chalisa Kallayanamitra, Andrew ID: ckallaya

ACS Data Collection and ETL Process & Rationale

Process	Rationale/Note
Python	
1. Wrote Python code on Spyder to access the **2018 5-year Estimates ACS dataset** by importing the **censusdata** package. Notably, I requested and used the key from the U.S. Census to prevent "Too Many Requests" error. I wrote a code to access **both city- and county-level data**. And I mainly used the pandas library to manage the data.	• Importing the census package is the easiest and most efficient way to access the whole dataset.[1] • The **5-year Estimates** are the data for **all areas.** It is the **largest sample size** and the **most reliable** dataset among the other ACS datasets. So, **the 5-year estimates are the best available ACS data now**.[2] • I used pandas instead of ohio library because I'm more comfortable using this library and when I tried to download the data, it took only about a few minutes to get the data.
2. Prompted the users to choose the **State** and level of data **granularity**.	• This is to keep the data collection process **modular and generalizable** so we can do it for a different state and level of granularity.
3. Wrote the Python code to **download the data, select the variables** of interest and **rename** them. Then, added the columns to represent the block group/city.	• I selected the variables that could potentially help us predict the likelihood of individual voting turnout in general elections: **race/ethnicity, gender, and education** (18 variables altogether). This to understand the socioeconomic status of at the block group-level which has an implication to the individual turnout likelihood. • Adding the columns to represent the block group/city would also make each row **uniquely identifiable**.
4. **Save the data frame as a .csv** and then save the Python code as CensusData.py.	
DBeaver	
5. Copy the group_students_database and change the name to turnout1_database	• Since the connection and the ssh setups are the same for the group_students_database, it is easier to just duplicate the database and just change the database name. This is to **prepare the database before uploading the data into it.**
Terminal commands	
6. Run this script on local path: scp -i "C:\Users\{MY_ID}\.ssh\id_rsa" "{MY_.PY_PATH}" {MY_ID}@mlpolicylab.dssg.io:~/	• This script is to copy the Python file from my local computer to the class server.
7. Run: ssh ckallaya@mlpolicylab.dssg.io	• This script is to connect to the class database.
7. Run: source /data/groups/turnout1/dssg_env/bin/activate	• This is to connect to the team's shared python virtual environment (manually).
8. Run: python CensusData.py, then choose the State (here I'd choose 12 for Florida) and level of granularity	• This is to run the Python script and store the .csv file on the virtual environment.

[1] We can access the ACS data via other channels such as the API or the U.S. Census advance search tool.

[2] The 5-yaer Estimates usually provides the least current data, but currently the most current data for other dataset is 2018, which we can get them from the 5-year estimates now.

<table>
<tr><td>(I’d choose 2 for the block group data). Then Run: ls to check if the files are stored successfully.</td><td>• Look for the file named data_county_cleaned.csv</td></tr>
<tr><td>9. Run: head -n 1000 data_county_cleaned.csv | tr [:upper:] [:lower:] | tr ' ' '_' | sed 's/#/num/' | csvsql -i postgresql --db-schema socioecon_schema1 --tables socioecon_table1</td><td>• This is to transform the column names of the variables into the format that Postgres prefers.</td></tr>
<tr><td>10. Wrote sql script using command vi followed by the name of the file and .sql to create a schema and a table for the ACS data. Here, I set the role (I checked the role in DBeaver group_students_database), name the schema and the table, and list all the variables, their types (most are decimal) and “not null”. Finally, copy the schema and table from the data_county_cleaned.csv.</td><td>• The intention of writing this sql script is to allow the script to be run with a single command without doing it manually line-by-line.
• Note: I used IF NOT EXISTS and DROP TABLE IF EXISTS options for the schema and the table to create the new schema only if it does not exist and check if the table exists prior to the dropping of the table. This should prevent an error.</td></tr>
<tr><td colspan="2"><b>Postgres</b></td></tr>
<tr><td>11. Run: psql -h mlpolicylab.db.dssg.io -U ckallaya turnout1_database -f data_county_cleaned.sql</td><td>• This is to run the data_county_cleaned.sql script on the turnout1_database.</td></tr>
<tr><td colspan="2">12. Went back to the DBeaver and saw that the <b>socioecon_schema1 and the socioecon_table1</b> was there with 11,442 rows, consistent with the number on google.</td></tr>
</table>

Location of the database table: In turnout1_database, look into socioecon_schema1 > Tables > socioecon_table1

Github link: https://github.com/ckallaya/Data-Collection-and-ETL