Team name: “Team 7 Phil”

Team Members: Damiso Hutchinson, Harsha Vinoy, Corey Anderson, Matthew Kenney, Eric Shadle

Project Title: “Essential Database for Bootcampers” (EDB)

Description: The goal of this project was to create a useful geodatabase (at the level of US states) for boot campers looking for data science or analytics jobs in the USA. We combined five data sources into a relational database (postgreSQL) with five tables: **housing**, **ds_jobs**, **breweries**, **salary**, and **state_boundary**. We have provided a notebook file with an example workflow in Python using sqlalchemy, pandas, geopandas, and matplotlib to query the database and make a corresponding map of the output (see example map, below).

- All tables contain the column ‘state’ with the standard two-letter, uppercase acronym for US state (e.g., ‘CA’ for California and ‘GA’ for Georgia).
    - The **housing** and **state_boundary** tables each have 51 rows, representing the 50 US states plus the District of Columbia. In other columns with ‘state’, there is a many-to-one relationship with rows in the ‘state’ column of **housing** (and **state_boundary**).
- The **state_boundary** table contains a geometry column containing the wkt-formatted vector boundary.
    - After importing from postgreSQL, you can transform the DataFrame back into a GeoDataFrame by re-assigning the shapely geometry (where df is a placeholder for the name of your DataFrame containing the geometry column):

        > df['geometry'] = df['geometry'].apply(wkt.loads)
        > gdf = geopd.GeoDataFrame(df, geometry = 'geometry')

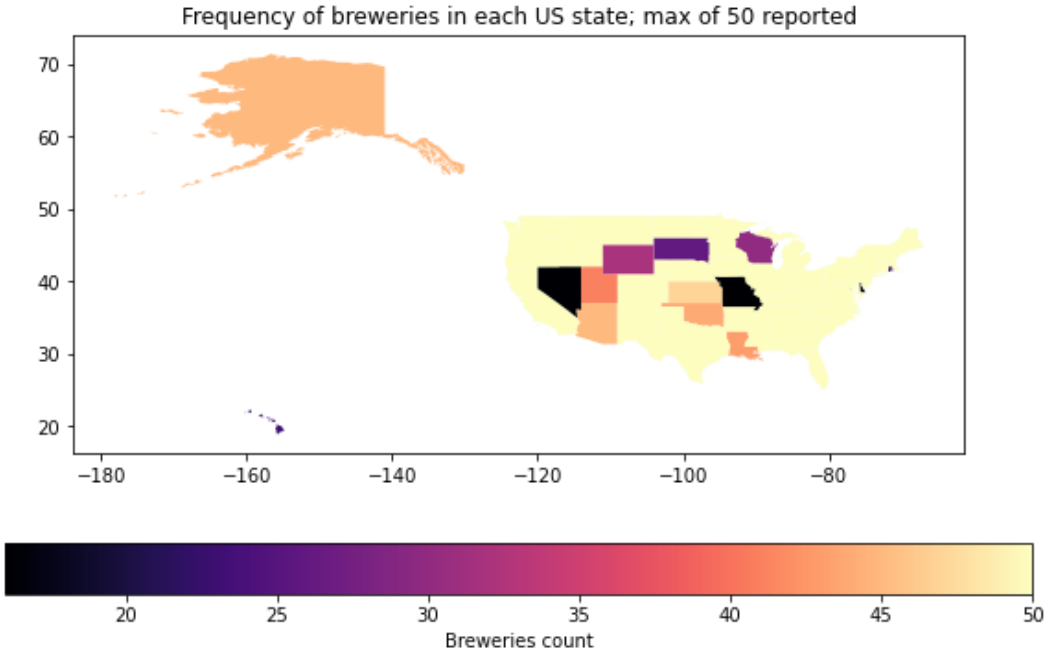

Table of Contents:

housing:

- cleaning_house.ipynb
- output_csv/output_housing_values_by_state.csv
- cleaner_housing.ipynb
- output_csv/avg_housing_cost.csv # Use this for SQL table.

ds_jobs:

- job_postings_data.ipynb
- output_csv/clean_job_data.csv
- job_cleaner.ipynb
- output_csv/cleaner_job_data.csv # Use this for SQL table.

breweries:

- breweries_5050_noSQL.ipynb # No SQL example for these records
- breweries_5050_to_table.ipynb
- ouput_csv/dirty_breweries.csv # Use this for SQL table.

salary:

- Salary_Data_Transformation.ipynb
- output_csv/clean_salary_data.csv
- wage_cleaner.ipynb
- ouput_csv/cleaner_salary_data.csv # Use this for SQL table.
-

state_boundary:

- geopandas_states.ipynb
- output_csv/state_poly.csv # Use this for SQLtable.

For query session via Python with geopandas examples

- Alchemist.ipynb

Data sources:

- **Table: housing**:

Source URL:

- https://www.zillow.com/research/data/

Associated files:

- cleaner_house.ipynb
- output_housing_value_by_state.csv
- dirty_zillow_homevalues_SFR_state.csv

Description:

- I went through Zillows website and found the data that we needed by state. Once I found the correct data that matches the purpose of our Project I downloaded it as a csv file. Once downloaded, I imported it into pandas for cleaning. Once cleaned down to state and average price for 2019 I simply outputted it as csv for import into SQL database.

Limitations/challenges:

- My first hurdle was missing, rows of data, probably due to a lack of record keeping for individual state. Luckily, record keeping and reporting did get better by the time the data reached 2019.
- Another hurdle I had was after cleaning sorting and outputting my final csv, my original dataset did not cover all 50 states as I thought. It only covered 40 out of the 50 states for some odd reason. Back to the drawing board.
- Once back on Zillow site I found another data set that matches our purpose a lot better, that did have data for all 50 states. Luckily all of my code was still able to work so all I had to do was import the new raw dataset and press play.

- **Table: ds_jobs**

  Source url:
  - https://www.kaggle.com/jobspikr/data-scientist-job-postings-from-the-usa

  File:
  - job_postings_2019.csv

  Associated files:

  - Job_postings_data.ipynb
  - output_csv/clean_job_data.csv
  - Job_cleaner.ipynb
  - output_csv/cleaner_job_data.csv

  Description:
  - The dataset contains 10,000 data scientists job postings in the US with different levels of experiences and requirements in 2019. The csv file from kaggle was used in pandas to complete the transform step.

  Limitations/challenges:
  - Job_title column values had multiple ways of describing the same job title. I had to change the names of job titles in order to be able to categorize the job titles values.
  - City and state columns had values such as computer or work from home.I had to convert these values to be called “remote” in the state and city columns.
  - Some of the values in the state column had zip code associated with it. I had to strip away the extra details from the values and only keep the state name.
  - With state columns, I had to convert the full state name to abbreviations of the state name in order to be able to merge with other datasets.
  - Some state IDs that were not keyed: ‘NN’, ‘PR’.
  - Had to make the dataframe SQL ready: na value is “NULL”, no oid, etc.

- **Table: breweries**

  Source API:
  - https://www.openbrewerydb.org/

  Associated files:
  -
  - breweries_5050_to_noSQL.ipynb
  - breweries_5050_to_table.ipynb
  - dirty_breweries.csv

  Description:
  - Used ‘requests’ in Python to call API and search for breweries by state.

  Limitations/challenges:
  - Record is noSQL; so each has to be converted to a row in a DataFrame.
  - Only 50 records can be returned in a single call (with no next page token to retrieve a;dditional pages). Getting all records in state would require a different scheme.
  - Had to make the dataframe SQL ready: na value is “NULL”, no oid, etc.

- **Table: salary**

  Source URL:
  - **https://www.bls.gov/oes/tables.htm**

  File:
  - salary_data_2019.csv

  Associated files:

  - Salary_Data_Transformation.ipynb
  - output_csv/clean_salary_data.csv
  - wage_cleaner.ipynb
  - ouput_csv/cleaner_salary_data.csv # Use this for SQL table.

  Description:
  - The original data source was a XLS file containing approximately 36,000 rows of the average job salaries across the different industries in the U.S. We used pandas to clean the data to only represent salary information related to the data science industry.

  Limitations/challenges:
  - The column labels were difficult to interpret so they were renamed to make them easier to understand.
  - To be compatible as a foreign key, a new column was created to contain the state abbreviations in place of the state's full name. A dictionary and the mapping function was used to accomplish this.
  - There were only two rows in each state related to data science so all other industries needed to be dropped.
  - Only four columns were relevant so all other columns were dropped.

- **Table: state_boundary**

  Surce_url:
  - https://hub.arcgis.com/datasets/CMHS::states-shapefile/about

  File:
  - shapefiles/States_shapefile.shp

  Associated files:
  - geopandas_state.ipynb
  - state_poly.csv

  Description:
  - Imported shapefile with polygons into geopandas; then exported as .csv for postgreSQL.

  Limitations/Challenges:
  - Geometry has to be re-assigned before it can be converted back into a geodataframe.
  - Having multiple copies of a polygon for a state would not be efficient; effect of plotting subsets of polygons needs to be explored.