ETL PROJECT FINAL REPORT

Tracy, Amber, & Ashley

1. Objectives

The purpose of this project was to extract, transform, and load several sources of data using tools that were learned in the Data Analytics Course. Our team's ETL project focused on environmental indicator data in different counties and across three decades, particularly greenhouse gas (GHG) emissions and climatological disasters. The goal of the project was to have a combined dataset that could possibly link countries with higher GHG emissions with increases in climatological disasters over time.

2. Extraction

Data was selected and retrieved from: https://www.kaggle.com/ruchi798/global-environmental-indicators. Two CSV files were downloaded from this site: GHG Emissions and Climatological Disasters.

The datasets included the following attributes:

GHG Emissions: 'CountryID', 'Country', 'Time Series - Greenhouse Gas Emissions (GHG) total without LULUCF, in 1000 tonnes of CO2 equivalent' (1990-2018 & Latest Year), 'GHG total without LULUCF, latest year' (1000 t of CO_2 equivalent), '% change since 1990', 'GHG emissions per capita, latest year' (kg of CO_2 equivalent)

Climatological Disasters: 'CountryID', 'Countries or areas', 'Occurrence 1990-1999', 'Occurrence 2000-2009', 'Occurrence 2010-2019', 'Total deaths 1990-1999', 'Total deaths 2000-2009', 'Total deaths 2010-2019', 'Persons affected 1990-1999', 'Persons affected 2000-2009', 'Persons affected 2010-2019'

3. Transformation

Both CSVs were loaded into two different jupyter notebooks and cleaned with pandas. The final dataframes were exported into CSVs.

Greenhouse gas emissions CSV

Through the use of Pandas, the GHG CSV was loaded into a jupyter notebook. At first the columns were called "Unnamed: 0", 'Unnamed: 1', '1990', '1991', '1992', '1993', '1994', …'2018', 'Latest Year', '1000 t of/n CO2 equivalent', '%', and 'kg of CO2 equivalent'. The first two columns were renamed to "CountryID" and "Countries or areas". The dataset was then unpiovoted using 'melt' to better analyze the data. As a result, the columns then became 'CountryID', 'Countries or areas', 'Latest Year', '1000 t of\n CO_2 equivalent', '%', 'kg of CO_2 equivalent', 'Year', and 'GHG emissions per 1000t'. The columns were then rearranged and the "year" data was binned by decade (1990-1999, 2000-2009, 2010-2019). The average greenhouse gas emissions for each decade was calculated and then inserted into a new "Decades" column. Finally, the dataframe was saved and exported as a .csv file.

Disasters CSV

The Climatological Disasters data was transformed using Pandas, unpivoting the data using 'lreshape', and using the 'cycle' function from 'itertools'.

After reading the csv as a DataFrame, we used 'lreshape' to unpivot and make the data easier to analyze with a tabular structure. We used 'lreshape' on this csv instead of the 'melt' method as we found this method addressed the unique attributes of this csv and produced the results we needed. Specifically, the Climatological Disasters csv included the decades as part of the column/header names (e.g. Occurrence 1990-1999), versus the GHG Emissions csv that contained columns of single years that could be binned into decades.

Next, we needed to sort the data in a manner that maintained the order accuracy of the 'Occurrences', 'Total Deaths', and 'Persons Affected' decades. We added the index name 'id' to sort by both 'Countries or areas' and 'id' which accomplished this.

Each country or area had an attribute/measure for 'Occurrences', 'Total Deaths', and 'Persons Affected', in decades '1990-1999', '2000-2009', and '2010-2019'. To be able to join the Climatological Disaster data with GHG Emissions, we added a decade column and used the cycle method to insert decades '1990-1999', '2000-2009', and '2010-2019' every three rows.

Finally, we renamed the columns to match the column names in the PostgreSQL tables (and to read correctly in PostgreSQL/pgAdmin), replaced '…' values to '0' to allow for calculations/analysis across the various measures, and pushed the transformed DataFrame to a new csv.

4. Loading

A PostgreSQL database was created in PGAdmin4 and set up on a cloud-based server using AWS so all group members could access the same database. Two tables were created in the database: "ghg" for holding the greenhouse gas emission data, and "disasters" for the disasters data with both tables having "CountryID" assigned as their primary key. Using SQLAlchemy in a jupyter notebook (Load.ipynb), both cleaned CSV files were loaded into their corresponding tables by establishing an engine and connection to the cloud-shared database. We used an if_exists='replace' function in the .to_sql loading code instead of 'append' in order to eliminate column name discrepancies between PGAdmin and the dataframes. The table column names that were used to join the two tables were renamed to include underscores to replace spaces and this also removed any formatting issues from the data being added to the database with the if_exists='replace' function. The two tables were queried by a 'SELECT DISTINCT' call joining on "CountryID" with a 'WHERE' statement signifying matching "decade" values. The table was ordered by the country name and decade so that the resulting table displayed the data alphabetically by country name and in chronological order. The pulled data showed the average greenhouse gas emission per decade along with the total number of disaster occurrences, total deaths from these disasters, and number of people affected. The queries and altering done in PGAdmin was saved as a .sql file and is with the other files for this project on github.