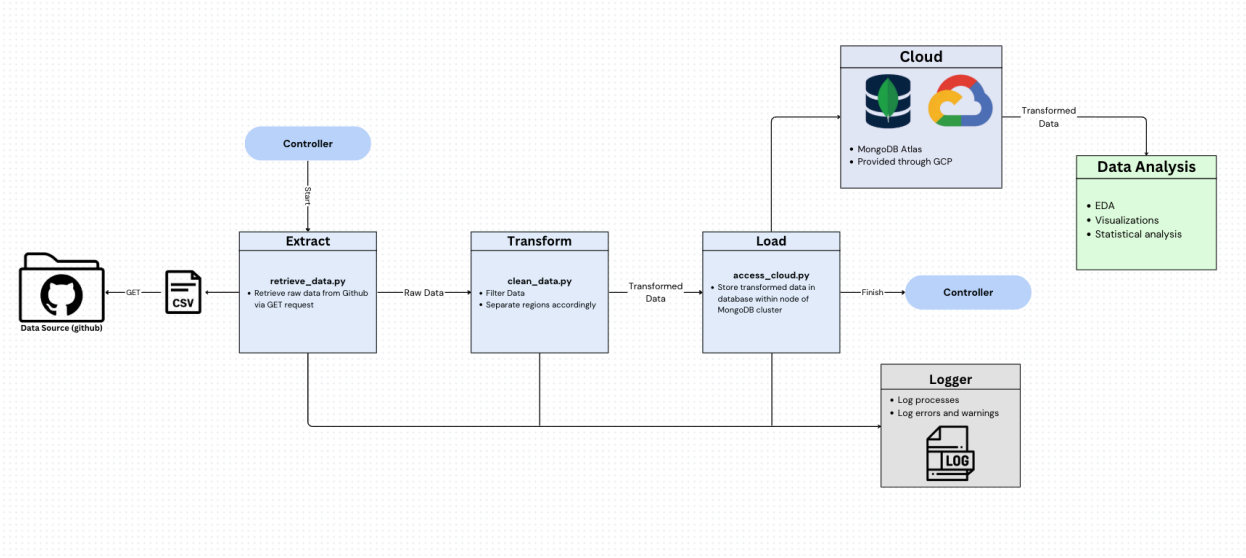
DS 2002 Final Project Report
Benjamin Yeh, Jaelyn Do, Jonah Lee, Kevin Lam

Data Selection and Exploration

Our datasets, sourced from GitHub, provide comprehensive insights into greenhouse gas emissions across various regions of the world. These include detailed data for individual countries (and their territories), continents, and diverse socioeconomic groups categorized by income levels.

ETL Setup

The ETL process consists of extraction, transformation, and loading and the ETL pipeline of our project was carefully designed to properly pull the required data, clean it, and store it for use. The ETL pipeline begins by pulling raw CSV data from our source on Github. The CSV data is then read into a Pandas dataframe and run through a Python script to begin the transformation process. In the transformation process, the data is cleaned, removing invalid values, reformatting certain values, and standardizing units. This step also filters the data, only keeping data from the years 1950 - 2023 because this year range was found to have the most readily available data. The data is also split into multiple datasets in the transform step. This is because in the 'country' column of the original dataset, it includes territories, continents, and socioeconomic classes. The transform script splits the data accordingly. Finally, in the loading step of the ETL pipeline, the data is stored into the cloud. Our choice for this was MongoDB through the Google Cloud Platform. Our data did not heavily rely on structure, nor did it have defined relationships, making a non-relational NoSQL database the better choice. Once the data is uploaded to the cloud, it is ready to be accessed for further analysis. Below is a diagram detailing the process of the ETL pipeline.



Analysis

From plotting populations of various countries against their respective carbon dioxide emissions, we found that there was a clear positive relationship between the two variables. Generally, as the

populations increased, carbon dioxide emissions increased as well. This relationship is supported by the R^2 value, which we calculated out to be 0.77, indicating that 77% of the variation in carbon dioxide emissions can be explained by the population. We speculated that the 23% of variation could be due to various reasons. The non-linearity in the lower range of the plot might suggest low industrial activity or better environmental policies, leading to lower emissions. As for the countries with higher emissions in the lower ranges, we speculated that this might be due to the fact that some small-population countries have high per-capita emissions. Despite these non-linearities in the lower range of the graph, the data suggests that carbon dioxide emissions scale more predictably at larger populations, likely due to industrialization and urbanization.
The R squared between CO2 and GDP was 0.87, which is very close to one. There was a log transformation done on these two variables, however, because based on the scatterplot, there was not a linear relationship. Our confidence interval for R squared was also very narrow, even with a high confidence level of 95%, so there is strong evidence that the true R squared value is very close to one.

Cloud Storage

The setup of cloud storage involved using Google Cloud Platform (GCP) and MongoDB Atlas. Due to the nature of the data, a NoSQL database was preferred. First, through MongoDB Atlas, a cluster was created with GCP as the provider. This is where the database/data is stored. In order to connect to the database, a Python driver was used. Before using Python to connect to the database, the access credentials were set up. An admin “Database user” was created, containing a username and password to access the database with read and write permissions. Using these credentials, a unique URI generated by Atlas can be used within Python to access the database. For security reasons, the URI was never hard coded into any publicly accessible code or within the repo itself. Instead the URI is stored in an environment variable, leveraging Python’s “dotenv” library and using a “.env” file to store the actual value of the URI. With the cluster, admin database user, and environment variables all set up, the Python scripts could access the database. In the ETL pipeline, the transformed data is uploaded to the database as collections. After the data is loaded, it is accessible via Python drivers and can be pulled and stored into dataframe through another script, making it readily available for use.