* **E**xtract: your original data sources and how the data was formatted (CSV, JSON, pgAdmin 4, etc).  
  + Five CSV files from World Bank data from Kaggle:
    - *GDP* by country by year from 1960 to 2016
    - *Population* by country by year from 1960 to 2016
    - *Fertility rates* by country by year from 1960 to 2016
    - *Life Expectancy* by country by year from 1960 to 2016
    - *Social Economic Score* by country by year from 1960 to 2016
* **T**ransform: what data cleaning or transformation was required.

Pandas Operations:

* + The original dataframe laid out the data with the “year” as columns. Reading the data for purposes of analysis would be difficult in this layout. To correct this issue, we used a melt function to transpose the year as rows by year and used the year as the index.
  + The original CSV file contained an empty row “Unnamed: 64” that had no values, we dropped that column.
  + Each variable used in the final dataframe initially came from different tables, tables had to be merged.
  + We used operations to make the GDP variable more manageable (it was originally expressed in scientific notation).
  + We used operations to create a new “GDP/Capita” column.
  + We experimented with different ways to import the database into Postgres (i.e. create the table first in the database or let Pandas automatically map the tables)

SQL operations:

* + We labeled variables using data types that would correctly format each column (e.g. proper rounding, decimals and the $ sign).
  + To create the final dataframe, we joined a table that was not part of the first couple of merges conducted in Pandas.
* **L**oad: the final database, tables/collections, and why this was chosen.
  + Our goal was to create a database that one could easily analyze/assess various economic indicators (or overall economic health) of each country by year. Toward this end, the final database contained an intuitive layout to show each economic indicator by year and country.
  + Postgres was chosen because our data set had an apparent relationship between ‘Year’ and ‘Country Name’. This relationship was easy to join in both Jupyter Notebook and Postgres, as demonstrated in our project.