**## Project Report**

At the end of the week, your team will submit a Final Report that describes the following:

\* **\*\*E\*\***xtract: your original data sources and how the data was formatted (CSV, JSON, pgAdmin 4, etc).

\* **\*\*T\*\***ransform: what data cleaning or transformation was required.

\* **\*\*L\*\***oad: the final database, tables/collections, and why this was chosen.

**ETL Project: Economic Freedom**

**Group members:** Sagi, Jake, Frances

**Project purpose**

The purpose of our project was to extract data that would enable us to answer the following questions:

How does the population density of a country affect the economic growth over the years?

How does the marginal tax rate impact a country’s economic freedom score?

**Extract**

Our data sources were:

[Countries of the World](https://www.kaggle.com/fernandol/countries-of-the-world) (<https://www.kaggle.com/fernandol/countries-of-the-world>) and

[Economic Freedom of the World](https://www.kaggle.com/gsutters/economic-freedom): (<https://www.kaggle.com/gsutters/economic-freedom>)

Both of the files were csv files. These files were read into our jupyter notebook.

**Transform**

The following steps were taken to clean the data:

One csv file had a trailing space after the name in cell; this prevented the merging of the two data frames on the Country key. The following code resolved this issue:

countrydf['Country'] = pd.core.strings.str\_strip(countrydf['Country'])

Several columns within the dataframe were renamed allowing the files to be merged and making the column headers more legible. The following code was used:

ecodf.rename(columns={'countries':'Country','ECONOMIC FREEDOM':'Economic freedom', '1d\_top\_marg\_tax\_rate':'Top marginal tax rate', '4c\_black\_market': 'Black market'}, inplace=True)

Several rows had missing data. To resolve this we selected the columns to be viewed for missing data. The following code was used:

ecodf\_copy = ecodf\_copy.dropna(subset=['year', 'Country', 'Economic freedom', 'rank','Top marginal tax rate'])

**Load**

The data that was extracted and transformed was moved to postgres database. A new database called as db\_world\_economy was created in local Postgres database.

For connecting to the database from Python, I utilized the SQL Alchemy module. The configuration parameters for the postgres database was stored in a local configuration file which was not checked in into Github using .gitignore.

Inside python utility, once successful connection was made, we dropped the below tables if they existed.

1. COUNTRIES: This table holds the reference data on the countries. The following datapoints were captured for the countries table.
   1. ID SERIAL PRIMARY KEY,
   2. COUNTRY VARCHAR(100) UNIQUE,
   3. POPULATION INTEGER,
   4. AREA INTEGER,
   5. POPULATION\_DENSITY FLOAT
2. COUNTRY\_ECONOMIC\_DATA : This table holds the actual economic data for individual country over the period of years. The following data was captured in this table.
   1. ID SERIAL PRIMARY KEY,
   2. COUNTRY\_ID INTEGER REFERENCES COUNTRIES(ID),
   3. ECON\_FREEDOM\_INDEX FLOAT,
   4. RANK INTEGER,
   5. TOP\_TAX\_RATE FLOAT,
   6. BLACK\_MARKET FLOAT

The tables were then re-created inside the utility to ensure that the staging table was empty before the loading process began.

The utility then iterated through each dataframe and extracted the data needed for insertion into the tables. Then a insert into the table was attempted. If the insert failed, the exception was caught and reported, and next row was picked up for processing. The utility kept track of the counts of various record statistics like

1. Total records processed
2. Total records successfully processed
3. Total records failed processing.

After the data was inserted the data was committed into the table.

One of the tests that I did was to load the data repeatedly. The expectation was that the data load would be successful first time but should fail with IntegrityException next time around because of the unique key constraint on the COUNTRIES table. I could not catch the IntegrityException in my exception block as I kept trying to catch PSYCOPG2 IntegrityException Exception. I learnt that the exception that was being thrown was from SQLAlchemy. I was able to successfully process the exception when I imported the IntegrityException from SQLAchemy.

After data loading was done, the data in the tables and the data in the dataframe were validated to ensure that no records were ignored in processing.