**DATE :** 10/15/2019

**ASSIGNMENT :** ETL-PROJECT

**TEAM MEMBERS :** Angeles Ramirez, Jon Hopkins, Brandon Steinke

1. **Extract Process**: ( The original data sources and how the data was formatted (CSV, JSON, pgAdmin 4, etc)

Our data was obtained from Kaggle.com. We attempted to aggregate several data sources for all countries of the globe. All data was downloaded as CSV.

----------------------------------------------------------------------------------

**File 1:**

**economic\_freedom\_index2019\_data.csv**

Kaggle source :

<https://www.kaggle.com/lewisduncan93/the-economic-freedom-index/downloads/the-economic-freedom-index.zip/1>

original source :

<https://www.heritage.org/index>

**File 2:**

**Life Expectancy Data.csv**

Kaggle source : <https://www.kaggle.com/kumarajarshi/life-expectancy-who>

LINK: <https://www.kaggle.com/kumarajarshi/life-expectancy-who>

------------------------------------------------------------ ----------------------------------------------------------------------------

**File 3:**

**country\_profile\_variablescsv**

Kaggle source : <https://www.kaggle.com/sudalairajkumar/undata-country-profiles>

LINK: <https://www.kaggle.com/sudalairajkumar/undata-country-profiles>

------------------------------------------------------------ ----------------------------------------------------------------------------

**File 4:**

**WHap\_2015.csv, WHap\_2016.csv, WHap\_2017.csv**

Kaggle source : <https://www.kaggle.com/unsdsn/world-happiness>

LINK: <https://www.kaggle.com/unsdsn/world-happiness>

1. **Transform Process:** ( the data cleaning or transformation was required.)

A lot of cleaning was required to create truly numeric values in our columns, commas and dollar signs were removed. N/A’s were replaced with zero values. And one off garbage content had to be systematically removed such as, removing everything but the number 6.1 from the following “ $6.1 CHF (2014 )”. Ensuring to convert all object columns to numeric int64 or float64 when necessary.

1. **Load Process :** (the final database, tables/collections, and why this was chosen.)

The majority of our work was performed in individual Jupyter files (per data set/ per person) and postgres used as the database module. We preferred to utilize the more familiar Postgres and SQL Alchemy method to create our tables and database rather than Mongo DB that we just learned. Angeles who setup the git repo, also established the primary database on her machine. Team members would Slack or git push their finished Jupyter files to the team, Angeles would download them her project folder and run all Jupyter cells to populate the database. Some initial issues arose when Data Frames that still contained dirty data or columns that did not match table fields would initiate errors. Also we made the error of pushing Data Frames directly to Postgres and allowing Pandas to bypass our custom tables and auto generate a table with dirty data. We attempted to export our Postgres database to our project folder without a password so anyone could run a query. Query files were create using…

**Database Name :** country\_db

**Table Names:**

* free\_ecos\_4
* life\_expectancy
* united\_nations\_transformed
* world\_happiness\_15
* world\_happiness\_16
* world\_happiness\_17