**ETL Final Project Report**

2Pandas:

* Lucy Hohler
* Giovanni Rigters

Does a country’s unemployment rate have an adverse effect on its Income group (low income, middle income, upper middle income, etc.)?

For this project we extracted two datasets from **Kaggle.com**

**Dataset 1: World Bank Youth Unemployment rates**

This dataset contains youth unemployment rates (modeled ILO estimate) Latest data available from 2010 to 2014

**Dataset 2: Poverty & Equity Database**

Latest poverty and inequality indicators compiled from officially recognized international sources. Poverty indicators include the poverty headcount ratio, poverty gap, and number of poor at both international and national poverty lines.

Inequality indicators include the Gini index and income or consumption distributions. The database includes national, regional and global estimates.

This database is maintained by the Global Poverty Working Group (GPWG), a team of poverty measurement experts from the Poverty Reduction and Equity Network, the Development Research Group, and the Development Data Group.

This is a dataset hosted by the World Bank. The organization has an open data platform found here and they update their information according the amount of data that is brought in. Explore the World Bank using Kaggle and all of the data sources available through the World Bank organization page!

**The ETL Process**

**Extract:** data sources used and how the data was formatted (CSV, JSON, MySQL, etc).

Our 2 datasets came from Kaggle and were both CSV formatted.

Dataset 1: World Bank Youth Unemployment rates

Dataset 2: Poverty & Equity Database

This data was loaded and read into Jupyter Notebook using the pandas dependency, specifically the ‘.read\_csv’ function

**Transform:** what data cleaning or transformation was required.

* Renaming and rearranging columns
* Dropping unnecessary columns
* Copying and loading data into DataFrames
* Transforming DataFrame to pull in the years as a column

For the first dataset we copied the data and dropped the first column called “Country Name”. This is not a column we needed since our dataset has a country code we could use.

Our data also had separate columns for years, so we had columns called: 2010, 2011, etc.

This is not the most efficient way to store data. If we needed to insert data for additional years we would constantly have to add new columns.

The solution to this problem was to write a for loop which iterated through the data and appended the country code, year, and rate (unemployment) to a new DataFrame.

So instead of having columns as years, we now have one column, called year, which contains the different years.

For the second dataset we copied the data, dropped the columns we did not need and renamed 7 of the columns.

**Load:** the final database, tables/collections.

We created one relational Database in MySQL, called world\_rate\_db, with two tables: unemployment, countries

While creating the countries table we used the country\_code as the Primary id. For the unemployment table we created a primary key column called ID which also contained a NOT NULL

We connected to the database in Jupyter Notebook by using create\_engine from the sqlalchemy dependency.

After checking to see which tables where in the database, we loaded the DataFrames into the tables using the ‘.to\_sql’ function.

Next, we queried the data from both tables using the SQL ‘.read\_sql\_query’ function.

We noticed that our unemployment table had two id columns, the default one and the ID column we created in MySQL. To solve this, we set the index of the DataFrame to the ID column.