**ETL Final Report**

**04-Jan-2020**

**Sruthi Karicheri**

**Contents**

1. Introduction ………………………………… 2
2. Data Extraction ………………………………… 3
3. Data Transformation ……………………………….. 3
4. Data Load ………………………………… 5
5. Summary ………………………………… 6

# Introduction

The project was carried out as a part of the Data Analytics course with the Rice University. Primary objective of the project is to demonstrate ability to extract data sets from multiple public project sets, transform them into a meaningful data structure and finally load into SQL server.

# Data Extraction

Data used in the project was extracted from two projects, a) *United States Energy, Census, and GDP 2010-2014*, and b) *US Oil and Gas Production June 2008 to June 2018*, in the Kaggle.com. Details of the data files from these two projects are given in Table-1.

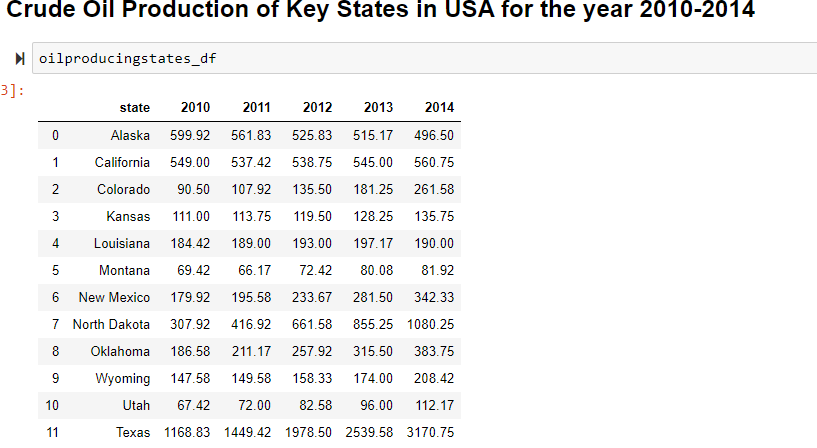
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S. No | Project | Data Type | Description | Format |
| 1. | United States Energy, Census, and GDP 2010-2014 | Economic Data Census and Geographic Data; | GDP for all the states of USA for the year 2010 to 2014 | .CSV |
| Oil and Gas consumption for all states for the year 2010 to 2014 |
| 2. | US Oil and Gas Production June 2008 to June 2018 | Energy Data | Oil and Gas production for all states for the year 2008 to 2018 | .CSV |

Table 1: Details of data source files that were extracted from different projects in the Kaggle.com

These data files in the .CSV format were extracted into *Pandas* data frame using *Jupyter* notebook.

# Data Transformation

*Crude Oil and Natural Gas Production Data:* The monthly crude oil and natural gas production for 2008 to 2018 were aggregated to yearly mean production for each state from the year 2010 to 2014 using *Pandas DateTimeIndex*, *Groupby() function* and filtered for 12 key oil producing states. In addition, the data frame was transposed, column renamed and reset the index.

Figure 1: Transformed crude oil and natural gas production data in Pandas.

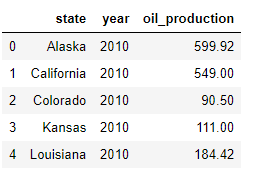
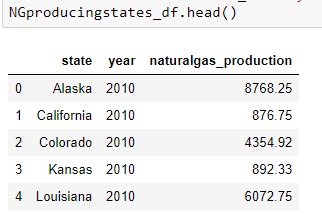
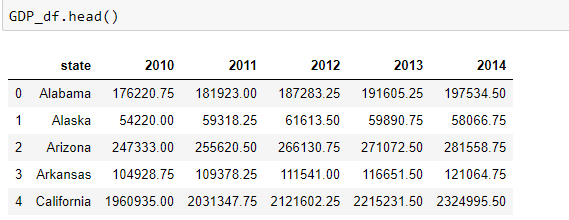
To make the analysis easier, the data was further normalized using Pandas *melt()* function for the above table. Followed by the normalization the year column was changed to integer type and the production data was defined as floating and rounded to two decimal points for better analytical access and accuracy respectively.

Figure 2: Normalized oil (left) and natural gas (right) production data

*GDP Data:*

The GDP data for year 2010 to 2014 was filtered out using the regular expression function from the Energy and GDP data. The quarterly columns were dropped keeping only the relevant yearly columns.

Figure 3: Transformed GDP data in Pandas.

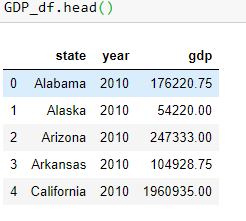
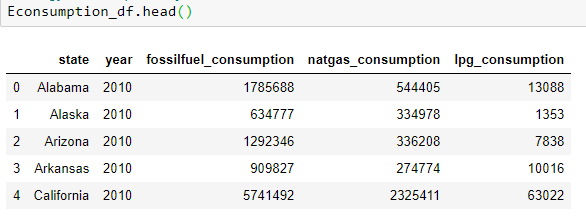
The data was further normalized using pandas melt() function for the above table and the isin() method was used to filter to the 12 key oil producing states.

Figure 4: Normalized GDP data

*Energy Consumption Data:*

The Fossil Fuel, Natural Gas and LPG consumption data were also filtered out from the Energy and GDP csv file and normalized for better analysis and to avoid redundancy of data, using the above functions and method used to get the GDP DataFrame. This data categories were selected for their association with the oil and gas production and GDP in the respective states.

Figure 5: Transformed Fossil Fuel, Natural Gas and LPG consumption data.

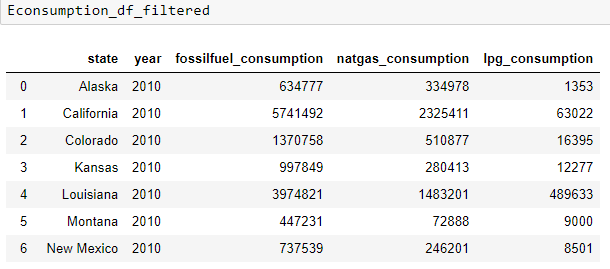


Figure 6: Normalized Fossil Fuel, Natural Gas and LPG consumption data.

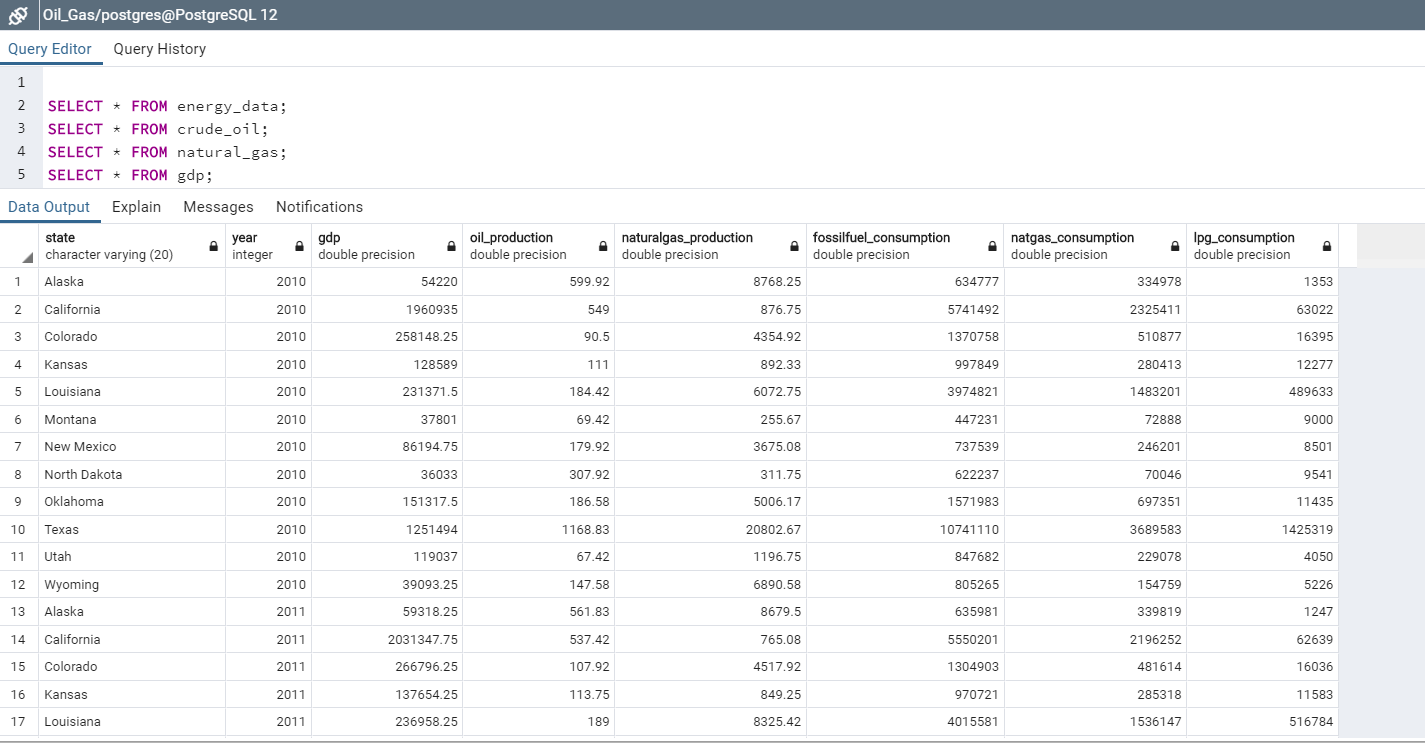
# Data Load:

The Jupyter notebook was connected to Postgres database and four tables were created in the database with the unique column names of the above data frames. State and year columns were assigned as Composite Key for each table.

The four dataframes containing, a) Crude Oil production data, b) Natural Gas production data, c) GDP data, and d) Energy Consumption data were loaded into tables in Postgres SQL database.

All the four tables were merged using INNER JOIN and keeping Composite Key as constraint. This allowed to verify the effectiveness of join table.

The decision to load the files into SQL, instead of MongoDB, was that the data collected from different files had to be joined in the database for analyzing the relationship between GDP, production and consumption rates of oil and gas in the key producing states of USA.

Figure 7: Final data table, combined from all the four individual tables loaded in SQL.

# Summary:

In this exercise, a successful demonstration of selecting multiple datafiles from correlated projects followed by their effective ETL process has been achieved. The extraction of data from multiple sources followed by multiple transformation steps including filtering, aggregating, normalizing and formatting were quiet challenging but essential for meaningful data preparation. The *Join* function used at SQL DB level was essential for combining the data sources to establish relationship and correlation among the aligned data categories.

ETL helps to prepare data for effective and efficient data base generation through multiple extractions, options for versatile transforms and loading of data avoiding redundancy and inconsistent dependency into the SQL DB.