Energy Trade Analysis Report – Nachiket Khambhati

Introduction

This is an Energy Trade Analysis data science project, whose dataset is available here: https://www.kaggle.com/datasets/unitednations/international-energy-statistics/.

Domain Understanding

I started off in the first week with the following research on the Domain Understanding, Data identification, and Data Collection aspects of the analysis.

Primary/Secondary Energy

Primary and secondary energy delineate the initial and refined stages of energy conversion, respectively. Primary energy sources are directly extracted from nature, including fossil fuels like coal and oil, renewable sources like sunlight, wind, and water, and nuclear fuels. These raw materials are foundational to energy production and must undergo conversion into more practical forms for consumption.

In contrast, secondary energy emerges from the transformation of primary sources into usable formats. Examples of secondary energy include electricity derived from coal, solar power, or wind turbines, as well as refined fuels such as gasoline and diesel. Secondary energy plays a pivotal role in powering homes, industries, and transportation systems, offering a more accessible and versatile energy form.

Renewable Energy Sources:

Renewable energy sources are sustainable, environmentally friendly alternatives that harness natural processes to generate power. Examples include solar energy, derived from sunlight through photovoltaic cells, wind energy produced by turbines capturing wind motion, and hydropower generated from flowing water. Biomass energy utilizes organic materials like wood or agricultural residues, while geothermal energy taps into the Earth's internal heat.

Conventional and Non-Conventional Energy Sources:

Conventional energy sources encompass traditional, widely used methods like fossil fuels (coal, oil, and natural gas) and nuclear power. Non-conventional sources, on the other hand, include renewable options alongside innovative technologies. While renewable sources fall under the non-conventional category, unconventional oil and gas extraction methods, such as shale gas and oil sands, also fit this classification.

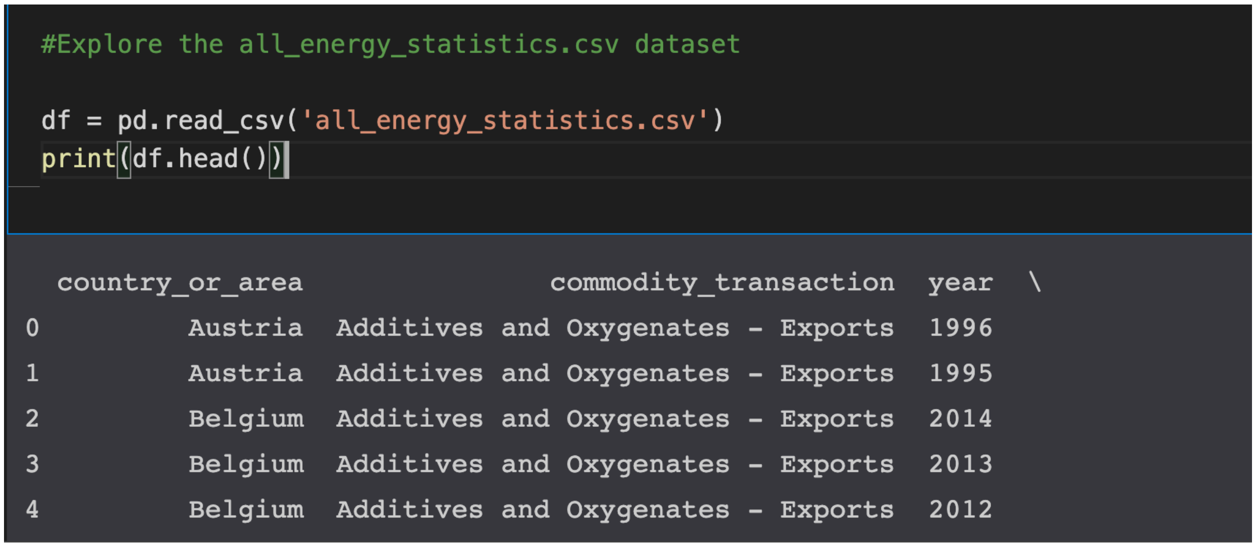
Methodology

Data Identification

The data for the energy trade analysis was provided by the United Nations statistics division on the UNData site. Available at: http://data.un.org/Explorer.aspx

Data Collection

The dataset was downloaded as a csv file and added to my Energy Trade Analysis repository as all_energy_statistics.csv. I used pandas to create a dataframe from the csv file as follows:

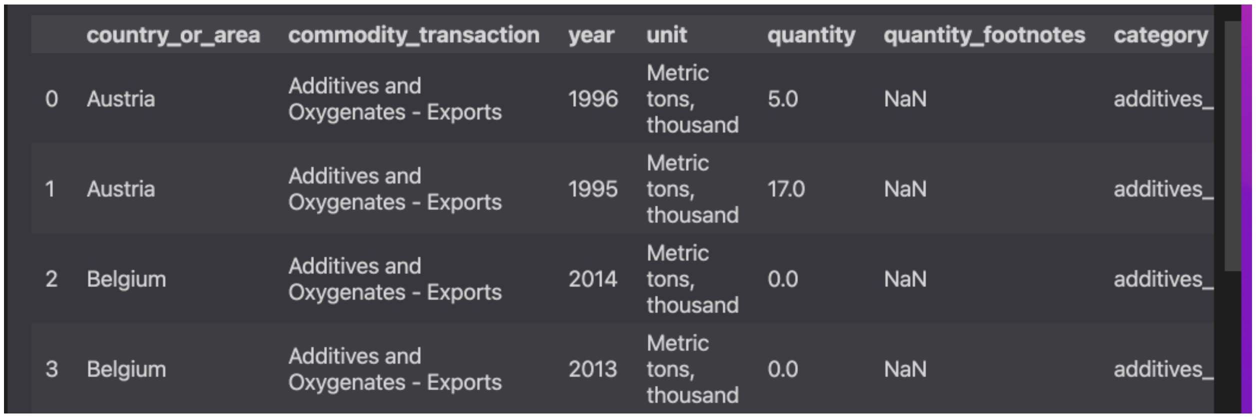


```
#Explore the all_energy_statistics.csv dataset

df = pd.read_csv('all_energy_statistics.csv')
print(df.head())
```

```
  country_or_area                 commodity_transaction  year  \
0         Austria  Additives and Oxygenates - Exports  1996
1         Austria  Additives and Oxygenates - Exports  1995
2         Belgium  Additives and Oxygenates - Exports  2014
3         Belgium  Additives and Oxygenates - Exports  2013
4         Belgium  Additives and Oxygenates - Exports  2012
```

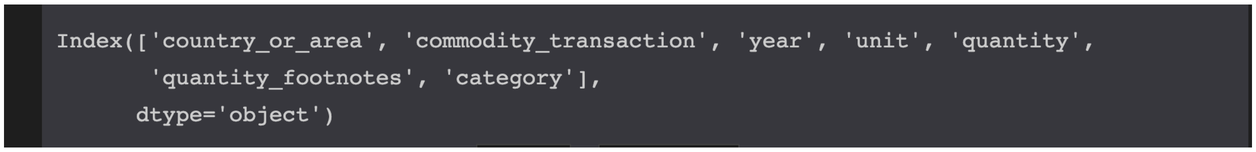
This would then facilitate in the next step which is to perform data cleaning and preprocessing.

Data Exploration



	country_or_area	commodity_transaction	year	unit	quantity	quantity_footnotes	category
0	Austria	Additives and Oxygenates - Exports	1996	Metric tons, thousand	5.0	NaN	additives_
1	Austria	Additives and Oxygenates - Exports	1995	Metric tons, thousand	17.0	NaN	additives_
2	Belgium	Additives and Oxygenates - Exports	2014	Metric tons, thousand	0.0	NaN	additives_
3	Belgium	Additives and Oxygenates - Exports	2013	Metric tons, thousand	0.0	NaN	additives_

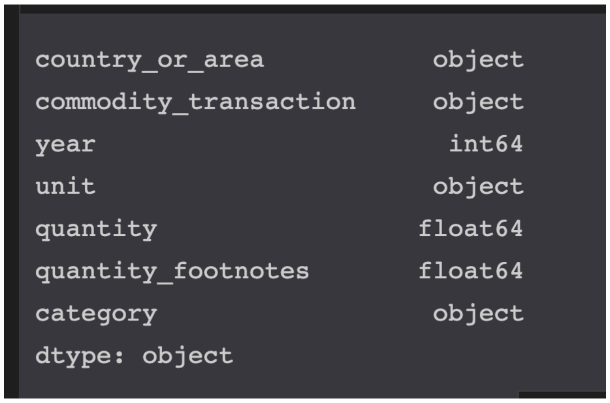
Based on this initial printout of the head of the dataframe obtained from the source csv, it appears that the relevant variables are:



```
Index(['country_or_area', 'commodity_transaction', 'year', 'unit', 'quantity',
       'quantity_footnotes', 'category'],
      dtype='object')
```

The only ambiguous one here is the 'quantity_footnotes' variable, which I don't know what it means and has the value 'nan' for at least the elements in the dataframe head.

Then I looked at the data types for each column:



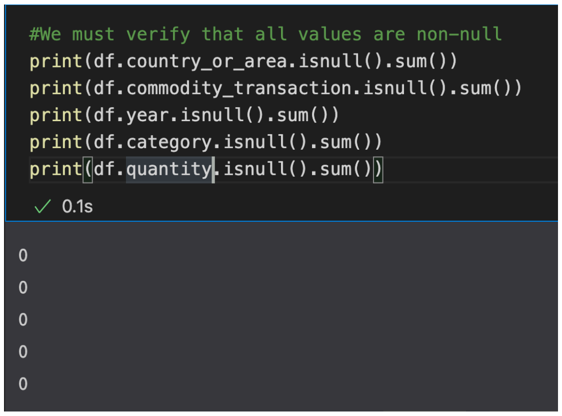
```
country_or_area           object
commodity_transaction     object
year                       int64
unit                      object
quantity                 float64
quantity_footnotes       float64
category                  object
dtype: object
```

So we have types 'object', int64 and float64

I proceeded to find the unique values of each categorical variable i.e. country_or_area, commodity_transaction, unit, category and quantity_footnotes. Upon doing this part of the exploration I also discovered that the quantity_footnotes variable is always either nan or 1.0, seemed not relevant to the analysis.

Data Cleaning

I am removing the quantity_footnotes column entirely as I don't think it will be useful for the analysis. Additionally, I need to explore for further inconsistencies in the dataset. This will begin by checking for missing data. First we have to verify that all values are non-null. This was done as follows:



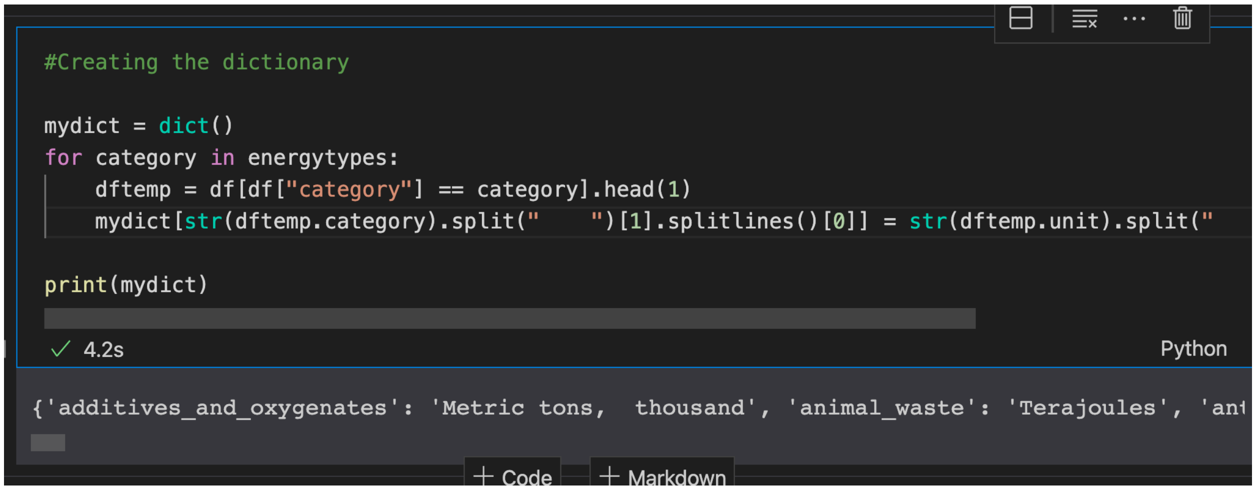
```
#We must verify that all values are non-null
print(df.country_or_area.isnull().sum())
print(df.commodity_transaction.isnull().sum())
print(df.year.isnull().sum())
print(df.category.isnull().sum())
print(df.quantity.isnull().sum())
```

✓ 0.1s

```
0
0
0
0
0
```

Data Preprocessing and Feature Engineering

Upon verifying this, we can continue preprocessing with categorical feature encoding, and feature engineering by numericizing quantity inputs through combining the unit and quantity variables. For now we are not differentiating between the export, import and other types of usage of energy, just going according to the energy category. I then created a dictionary that stores the unit associated with each energy type:



```
#Creating the dictionary

mydict = dict()
for category in energytypes:
    dftemp = df[df["category"] == category].head(1)
    mydict[str(dftemp.category).split("    ")[1].splitlines()[0]] = str(dftemp.unit).split("

print(mydict)
```

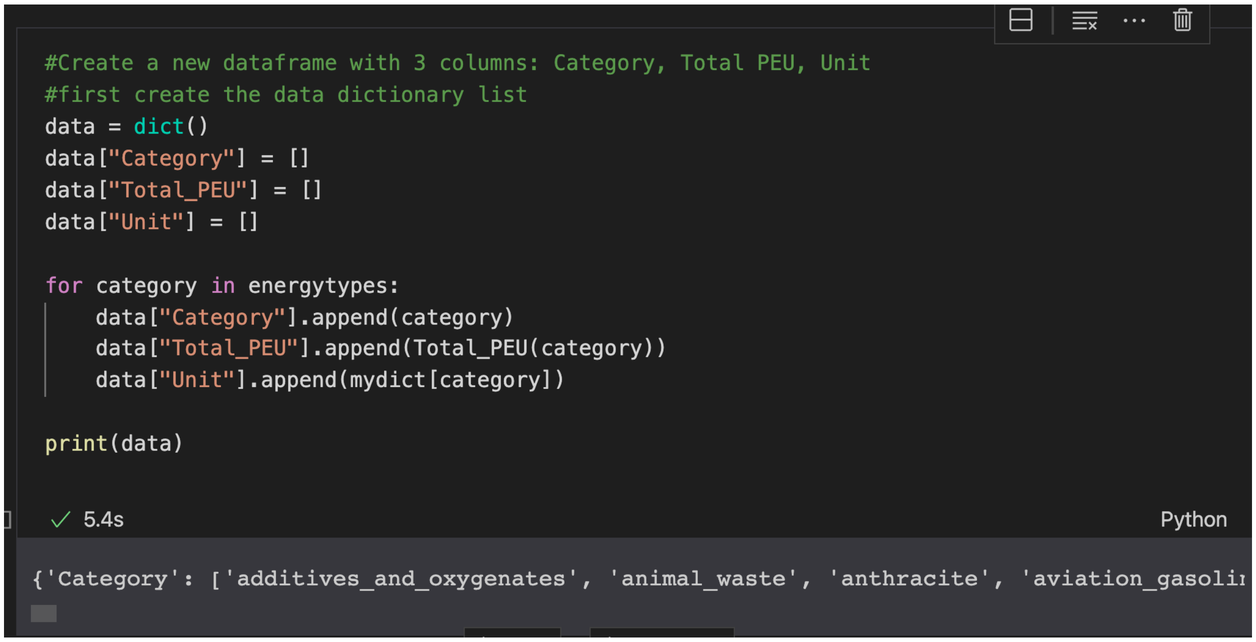
✓ 4.2s Python

```
{'additives_and_oxygenates': 'Metric tons,  thousand', 'animal_waste': 'Terajoules', 'an
```

Results and Analysis

<u>Part 1</u> – Total energy Production, Exchange, and Usage

The first goal is to obtain the statistics according to each energy type. We have the dictionary storing the unit associated with each type, now we have to create a new dataframe that stores the total energy production, usage, and exchange of each energy type along with the unit. First created a dict called data, which stored the category of each energy type and the total PEU (production, exchange and usage) of it along with the unit as follows:

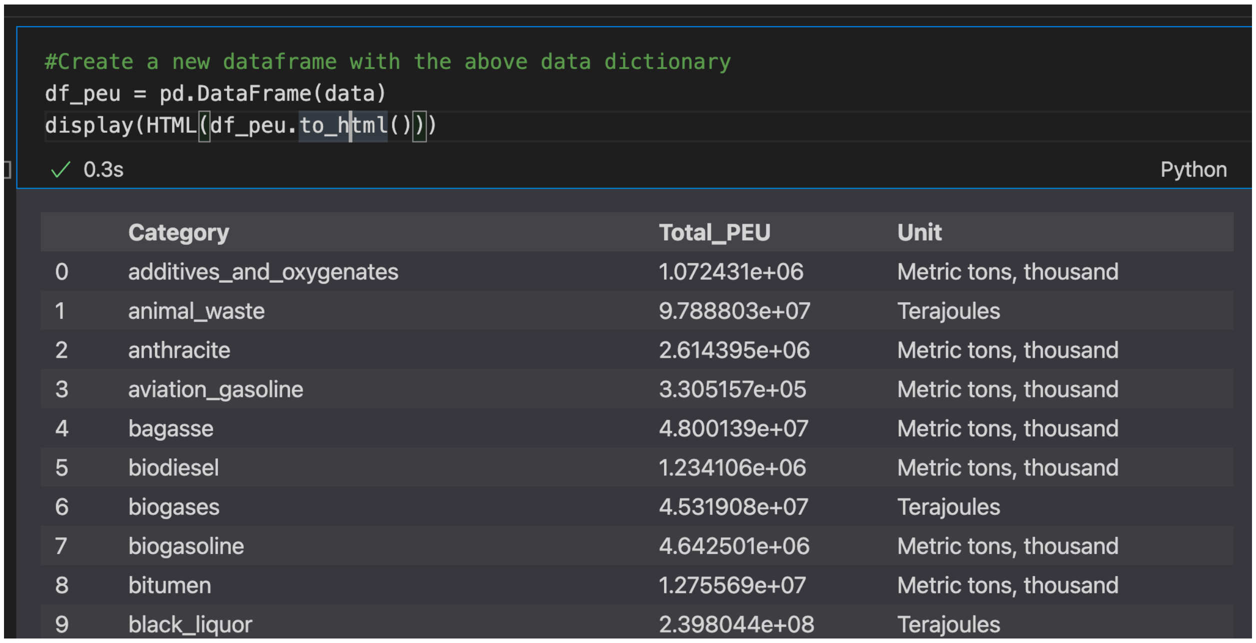


```python
#Create a new dataframe with 3 columns: Category, Total PEU, Unit
#first create the data dictionary list
data = dict()
data["Category"] = []
data["Total_PEU"] = []
data["Unit"] = []

for category in energytypes:
    data["Category"].append(category)
    data["Total_PEU"].append(Total_PEU(category))
    data["Unit"].append(mydict[category])

print(data)
```

✓ 5.4s Python

```
{'Category': ['additives_and_oxygenates', 'animal_waste', 'anthracite', 'aviation_gasoli
```

Next, I created a dataframe with columns Category, Total PEU, and Unit using this dictionary as follows: (This will be called a PEU-style dataframe from here on)



```python
#Create a new dataframe with the above data dictionary
df_peu = pd.DataFrame(data)
display(HTML(df_peu.to_html()))
```

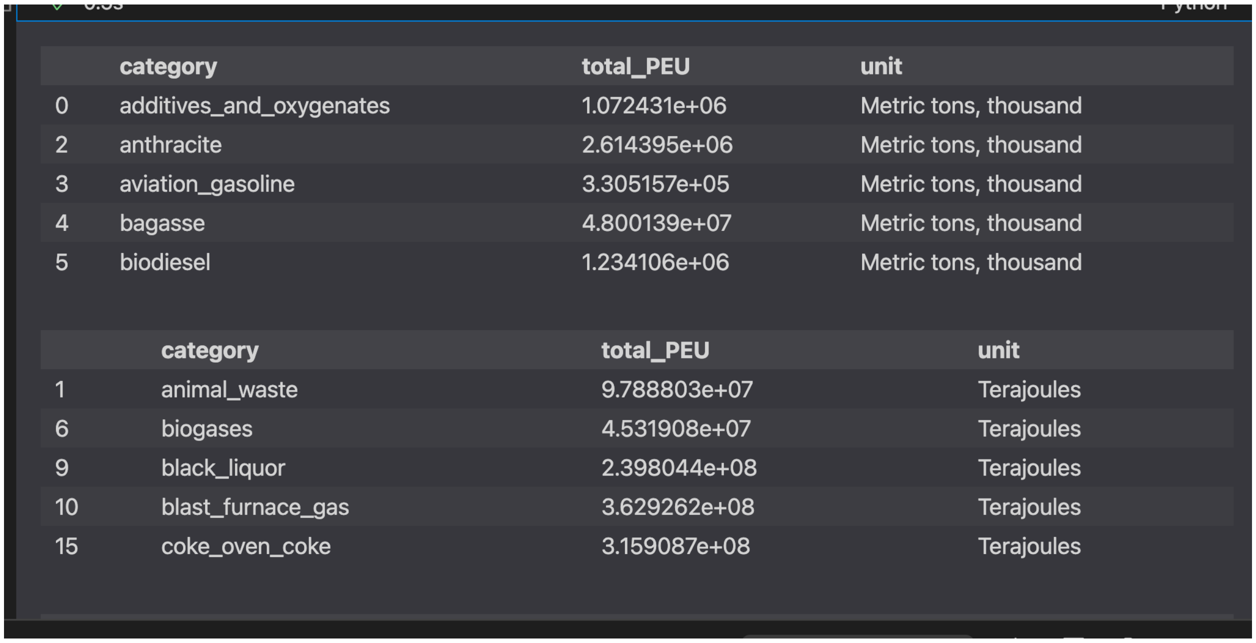
✓ 0.3s Python

	Category	Total_PEU	Unit
0	additives_and_oxygenates	1.072431e+06	Metric tons, thousand
1	animal_waste	9.788803e+07	Terajoules
2	anthracite	2.614395e+06	Metric tons, thousand
3	aviation_gasoline	3.305157e+05	Metric tons, thousand
4	bagasse	4.800139e+07	Metric tons, thousand
5	biodiesel	1.234106e+06	Metric tons, thousand
6	biogases	4.531908e+07	Terajoules
7	biogasoline	4.642501e+06	Metric tons, thousand
8	bitumen	1.275569e+07	Metric tons, thousand
9	black_liquor	2.398044e+08	Terajoules

This will have to be graphed and then analyzed, which will complete the first step of data analysis.

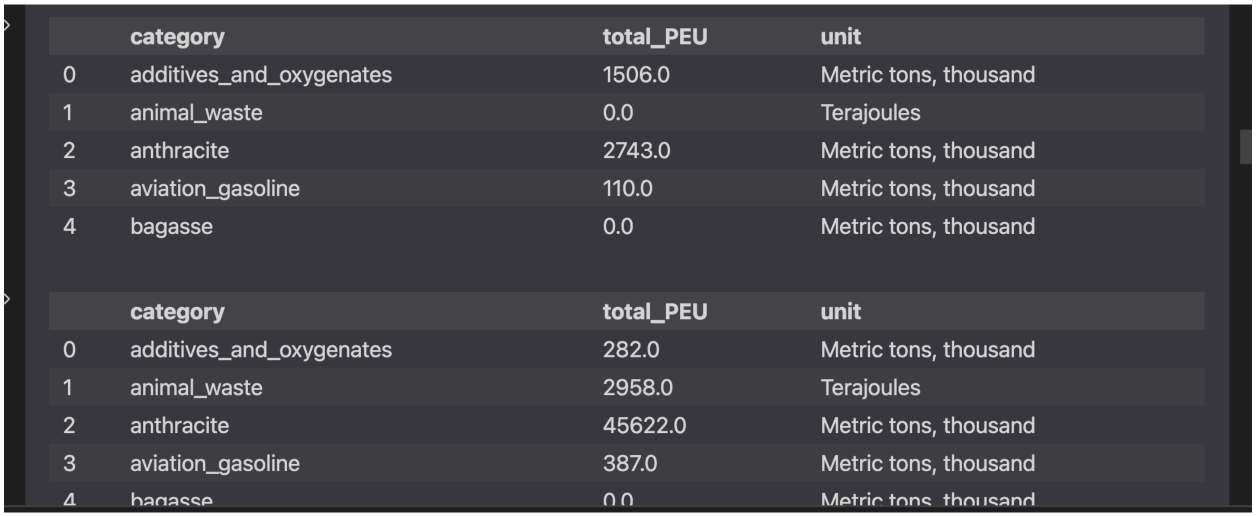
Part 2 - Comparative Analysis:

We will try to group the energy PEU dataframe according to the unique units, as that is the only way to compare across different energy types. This is done as follows:



```
    category                  total_PEU       unit
0   additives_and_oxygenates  1.072431e+06    Metric tons, thousand
2   anthracite                2.614395e+06    Metric tons, thousand
3   aviation_gasoline         3.305157e+05    Metric tons, thousand
4   bagasse                   4.800139e+07    Metric tons, thousand
5   biodiesel                 1.234106e+06    Metric tons, thousand

    category                  total_PEU       unit
1   animal_waste              9.788803e+07    Terajoules
6   biogases                  4.531908e+07    Terajoules
9   black_liquor              2.398044e+08    Terajoules
10  blast_furnace_gas         3.629262e+08    Terajoules
15  coke_oven_coke            3.159087e+08    Terajoules
```

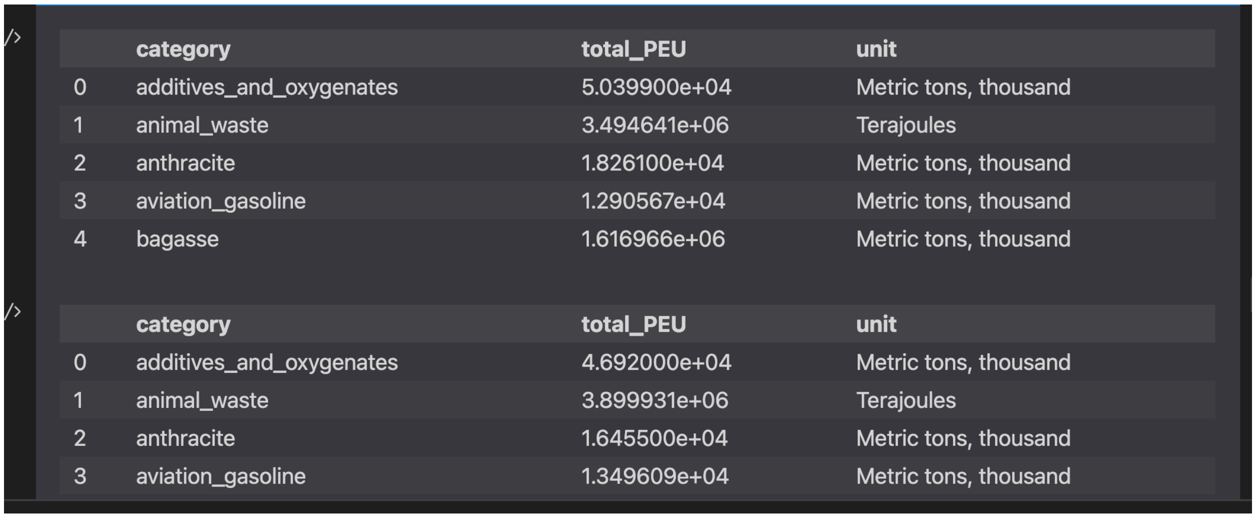
Now that this unit-dataframe dictionary is obtained, we move on to grouping by countries. This will be done by creating a PEU dataframe for each country in the list as follows:



```
    category                  total_PEU   unit
0   additives_and_oxygenates  1506.0      Metric tons, thousand
1   animal_waste              0.0         Terajoules
2   anthracite                2743.0      Metric tons, thousand
3   aviation_gasoline         110.0       Metric tons, thousand
4   bagasse                   0.0         Metric tons, thousand

    category                  total_PEU   unit
0   additives_and_oxygenates  282.0       Metric tons, thousand
1   animal_waste              2958.0      Terajoules
2   anthracite                45622.0     Metric tons, thousand
3   aviation_gasoline         387.0       Metric tons, thousand
4   bagasse                   0.0         Metric tons, thousand
```

This gives us a dictionary with country key and PEU style dataframe value.

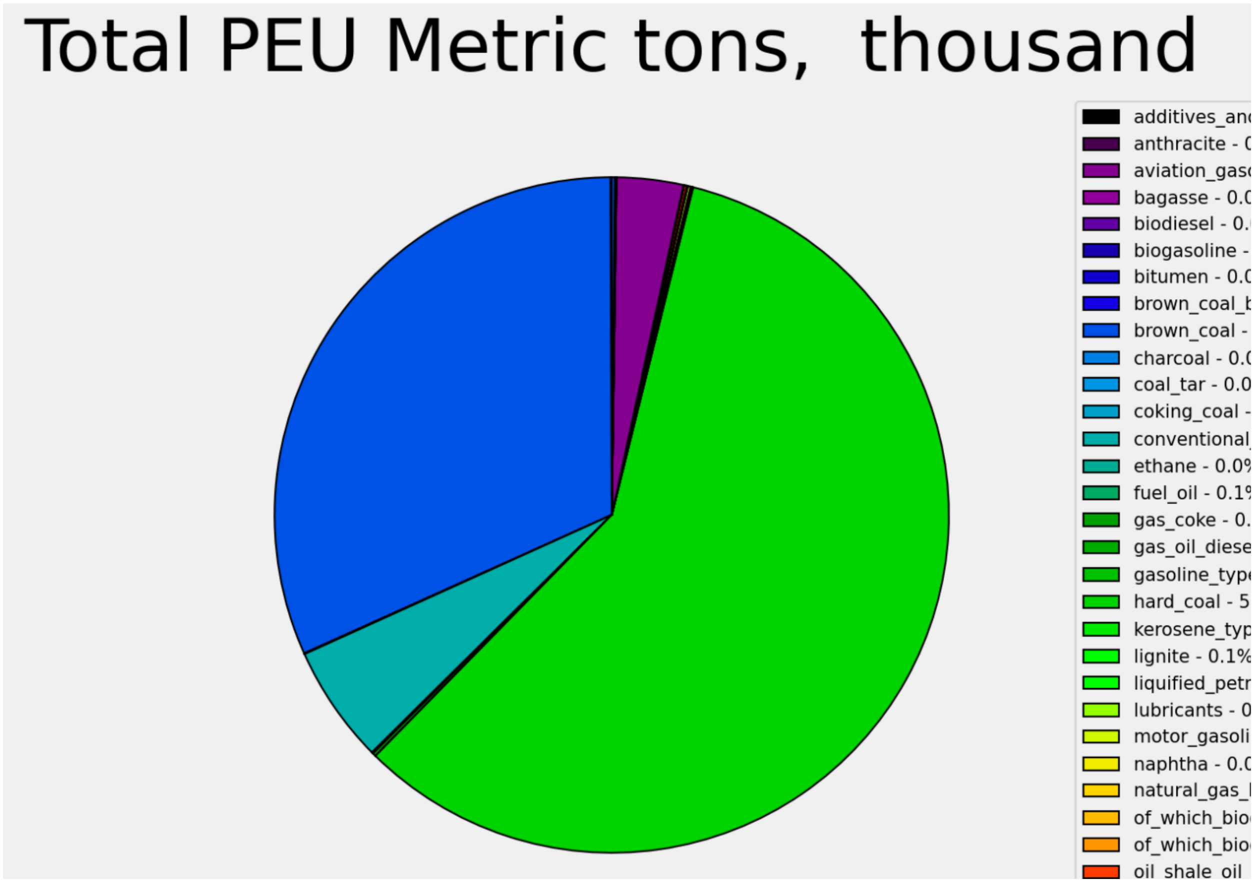
Same protocol was followed with countries as follows:



```
     category                  total_PEU       unit
0    additives_and_oxygenates  5.039900e+04    Metric tons, thousand
1    animal_waste              3.494641e+06    Terajoules
2    anthracite                1.826100e+04    Metric tons, thousand
3    aviation_gasoline         1.290567e+04    Metric tons, thousand
4    bagasse                   1.616966e+06    Metric tons, thousand

     category                  total_PEU       unit
0    additives_and_oxygenates  4.692000e+04    Metric tons, thousand
1    animal_waste              3.899931e+06    Terajoules
2    anthracite                1.645500e+04    Metric tons, thousand
3    aviation_gasoline         1.349609e+04    Metric tons, thousand
```

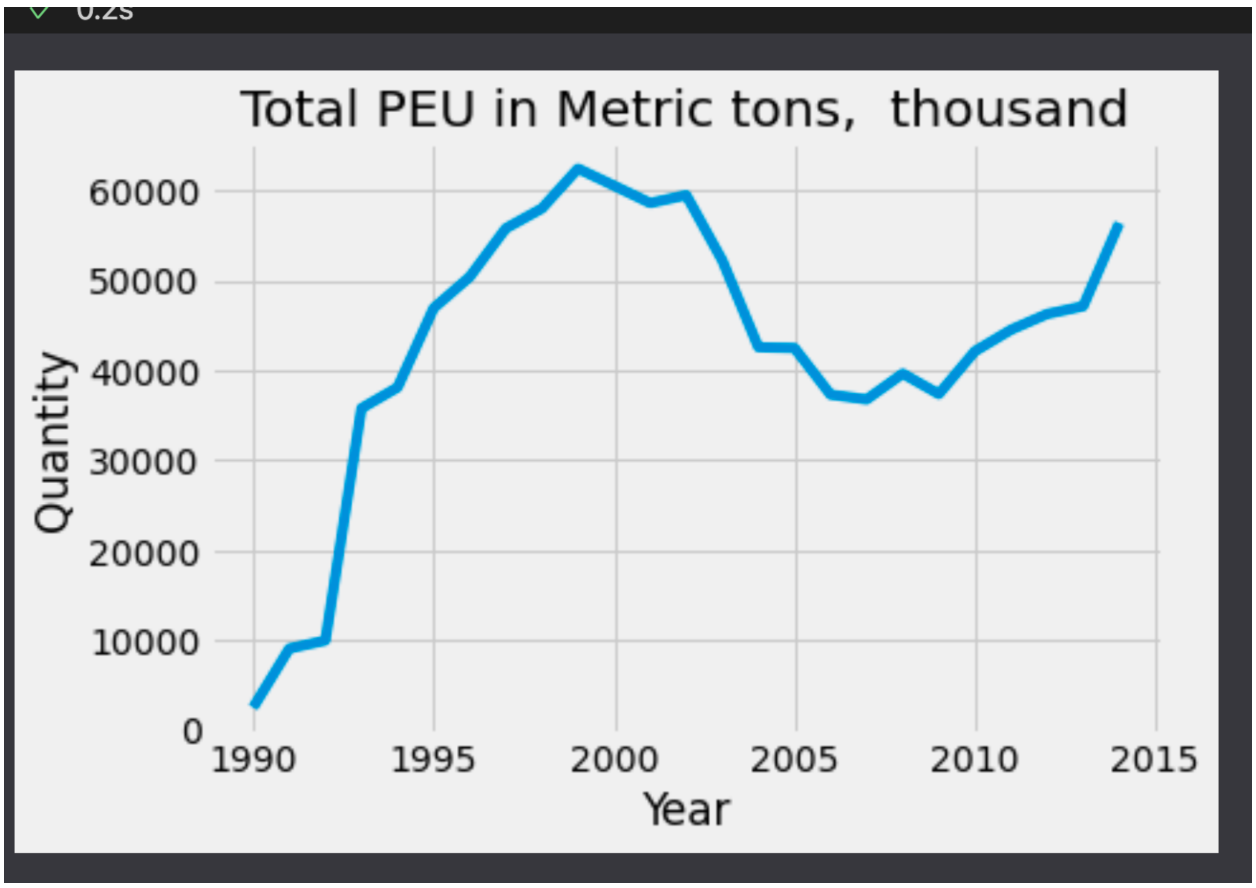
<u>Part 3</u> – Data visualization and analysis

The first part is visualizing total energy production. Only meaningful way to do it on pie charts is grouping the ones with same unit together as follows:

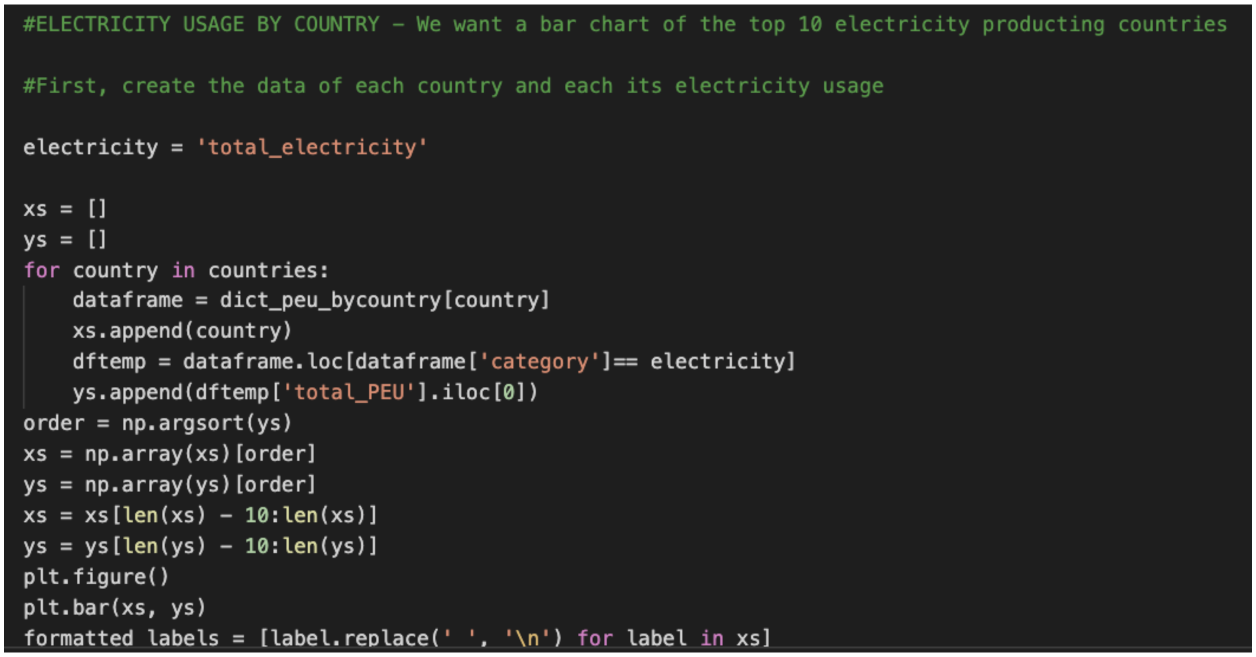


This was done for all units.

Next part is different countries and different years. The dataframes are ready. They need to be plotted. Here is an example for additives and oxygenates over the years:

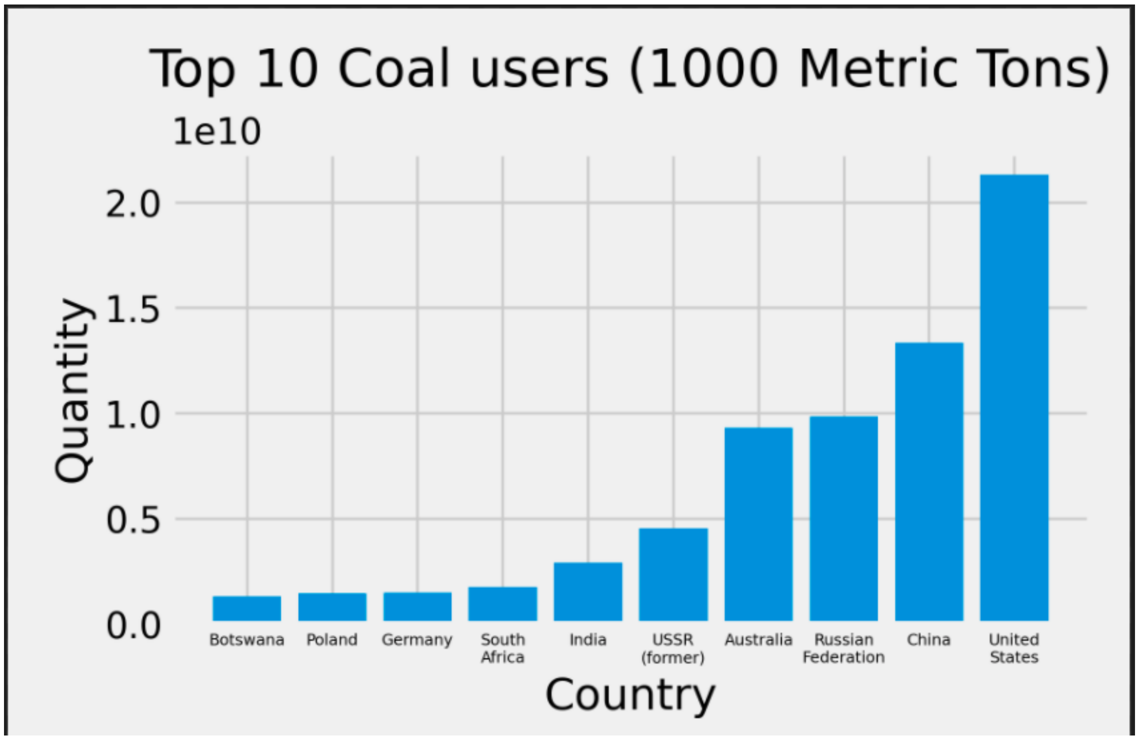


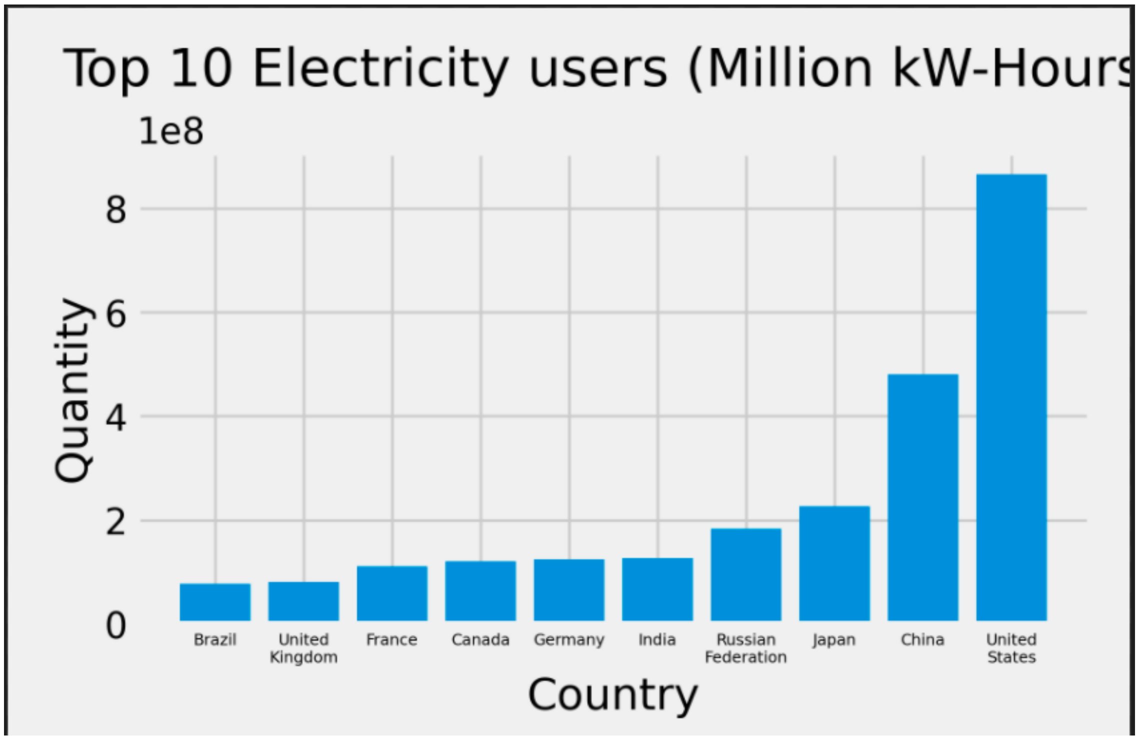
The next part is visualizing the data for the top 10 energy producing nations for each energy type. I chose to use only the top 3 energy types as indicated from the pie charts of total energy consumption by energy category. These are coal, electricity and natural gas. The following is a sample of the code snippet for electricity, which was repeated for coal and natural gas. I wanted to obtain the top 10 electricity, coal and natural gas producing nations for the total consumption from 1990 to 2015:

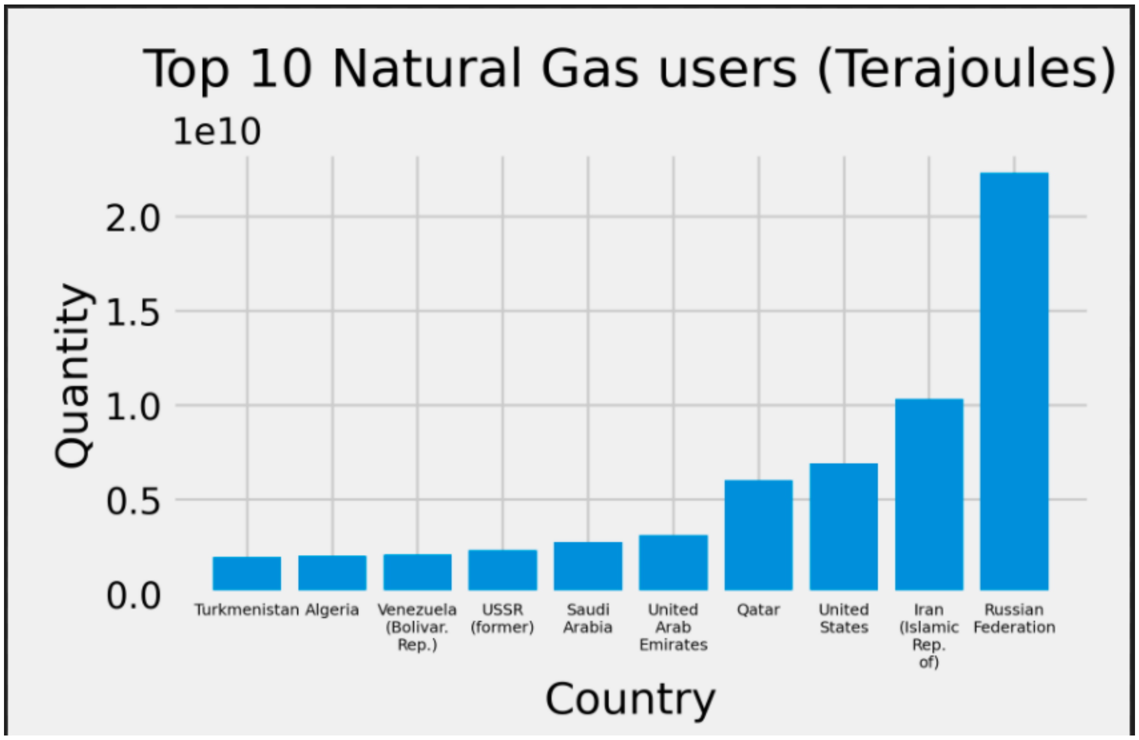


```
#ELECTRICITY USAGE BY COUNTRY - We want a bar chart of the top 10 electricity producting countries

#First, create the data of each country and each its electricity usage

electricity = 'total_electricity'

xs = []
ys = []
for country in countries:
    dataframe = dict_peu_bycountry[country]
    xs.append(country)
    dftemp = dataframe.loc[dataframe['category']== electricity]
    ys.append(dftemp['total_PEU'].iloc[0])
order = np.argsort(ys)
xs = np.array(xs)[order]
ys = np.array(ys)[order]
xs = xs[len(xs) - 10:len(xs)]
ys = ys[len(ys) - 10:len(ys)]
plt.figure()
plt.bar(xs, ys)
formatted_labels = [label.replace(' ', '\n') for label in xs]
```

This was repeated for coal and natural gas as well, and the following plots were obtained:




Top 10 Electricity users (Million kW-Hours
1e8
Quantity
0
2
4
6
8
Brazil
United Kingdom
France
Canada
Germany
India
Russian Federation
Japan
China
United States
Country


Top 10 Natural Gas users (Terajoules)
1e10
Quantity
0.0
0.5
1.0
1.5
2.0
Turkmenistan
Algeria
Venezuela (Bolivar. Rep.)
USSR (former)
Saudi Arabia
United Arab Emirates
Qatar
United States
Iran (Islamic Rep. of)
Russian Federation
Country