1 Achieving the Paris Agreement Goals: Projecting Energy Production Sources in the United States With Time Series Modeling

- Student name: Greg Osborne
- Student pace: self paced / part time
- Scheduled project review date/time: 4/28/23, 2:45 PM
- Instructor name: Morgan Jones
- Blog post URL: https://medium.com/@gregosborne (https://medium.com/@gregosborne)

2 EDA / Data Cleaning Notebook

This Jupyter Notebook is for EDA, Data Cleaning and exporting cleaned DataFrames. The data explored and cleaned in this notebook includes several more references and DataFrames than were included in the final project.

3 Data Sources

3.1 Energy Production / Consumption

- Energy Production Wattage by Country by Source: Our World In Data (https://ourworldindata.org/grapher/electricity-prod-source-stacked)
  - Original source: BP (https://www.bp.com/en/global/corporate/energy-economics/statistical-review-of-world-energy.html)
- Historic Electricity Demand by Country: Our World In Data (https://ourworldindata.org/grapher/electricity-demand?country=USA~GBR~FRA~DEU~IND~BRA)
  - Original source: BP (https://www.bp.com/en/global/corporate/energy-economics/statistical-review-of-world-energy.html)
- Energy demand projection up until 2050: IEA (https://www.iea.org/data-and-statistics/data-product/world-energy-outlook-2022-free-dataset)
  - Data sourced from the IEA's World Energy Outlook 2022 Report: Countries by Region (https://iea.blob.core.windows.net/assets/830fe099-5530-48f2-a7c1-11f35d510983/WorldEnergyOutlook2022.pdf#page=499)

4 Table of Contents

Project 5 – What would it take to Meet the Paris Agreement Energy Production Targets?

5 Python Libraries

In [1]:
```
# DataFrames and computation
import pandas as pd
import numpy as np

# To supress warnings
import warnings

# Setting DataFrame Display settings
pd.set_option("display.max_columns", None)

# For downloading files too big for GitHub
import requests
```

6 Loading the Data

6.1 Energy Production / Consumption

6.1.1 Historic Electricity Demand by Country

- Historic Electricity Demand by Country: Our World In Data (https://ourworldindata.org/grapher/electricity-demand?country=USA~GBR~FRA~DEU~IND~BRA)
  - Original sources:

- BP Statistical Review of World Energy. (https://www.bp.com/en/global/corporate/energy-economics/statistical-review-of-world-energy.html)
- Ember Yearly Electricity Data (2023). (https://ember-climate.org/data-catalogue/yearly-electricity-data/)
- Ember European Electricity Review (2022). (https://ember-climate.org/insights/research/european-electricity-review-2022/)

In [2]:
```
# Initial Report Scope
df_hist_dem_raw = pd.read_csv(
    "Data/electricity-demand.csv",
    encoding='ANSI')
df_hist_dem_raw
```

	Entity	Code	Year	Electricity demand (TWh)
0	Afghanistan	AFG	2000	0.57
1	Afghanistan	AFG	2001	0.69
2	Afghanistan	AFG	2002	0.79
3	Afghanistan	AFG	2003	1.04
4	Afghanistan	AFG	2004	0.99
...	...	...	...	...
5565	Zimbabwe	ZWE	2017	9.57
5566	Zimbabwe	ZWE	2018	10.21
5567	Zimbabwe	ZWE	2019	9.35
5568	Zimbabwe	ZWE	2020	9.58
5569	Zimbabwe	ZWE	2021	9.79

5570 rows × 4 columns

6.1.2 Energy Generation

- Energy Production Wattage by Country by Source: Our World In Data (https://ourworldindata.org/grapher/electricity-prod-source-stacked)
  - Original source:
    - BP Statistical Review of World Energy. (https://www.bp.com/en/global/corporate/energy-economics/statistical-review-of-world-energy.html)
    - Ember Yearly Electricity Data (2023). (https://ember-climate.org/data-catalogue/yearly-electricity-data/)
    - Ember European Electricity Review (2022). (https://ember-climate.org/insights/research/european-electricity-review-2022/)
  - See Ember's Data Methodology (https://ember-climate.org/app/uploads/2022/07/Ember-Electricity-Data-Methodology.pdf) document for definitions on the different electricity fuel sources.

In [3]:
```
# Initial Report Project 5 Scope
df_hist_elec_raw = pd.read_csv(
    "Data/electricity-prod-source-stacked.csv",
    encoding='ANSI')
df_hist_elec_raw
```

Out[3]:

	Entity	Code	Year	Other renewables excluding bioenergy (TWh) (zero filled)	Electricity from bioenergy (TWh) (zero filled)	Electricity from solar (TWh)	Electricity from wind (TWh)	Electricity from hydro (TWh)	Ele
0	Afghanistan	AFG	2000	0.0	0.00	0.00	0.0	0.31	
1	Afghanistan	AFG	2001	0.0	0.00	0.00	0.0	0.50	
2	Afghanistan	AFG	2002	0.0	0.00	0.00	0.0	0.56	
3	Afghanistan	AFG	2003	0.0	0.00	0.00	0.0	0.63	
4	Afghanistan	AFG	2004	0.0	0.00	0.00	0.0	0.56	
...	...	...	...	...	...	...	...	...	
10644	Zimbabwe	ZWE	2017	0.0	0.32	0.01	0.0	3.97	
10645	Zimbabwe	ZWE	2018	0.0	0.39	0.02	0.0	5.05	

6.1.3 Energy Demand Projection

- Energy demand projection until 2050: IEA (https://www.iea.org/data-and-statistics/data-product/world-energy-outlook-2022-free-dataset)

In [4]:
```
# Initial Report Project 5 Scope
df_proj_dem_raw = pd.read_csv(
    "Data/IEA-WEO2022_AnnexA_Free_Dataset_Regions.csv",
    encoding='ANSI')
df_proj_dem_raw
```

Out[4]:

	PUBLICATION	SCENARIO	CATEGORY	PRODUCT	FLOW	UNIT	REGION	YEAR	
0	World Energy Outlook 2022	Stated Policies Scenario	Energy	Total	Total energy supply	EJ	World	2010	
1	World Energy Outlook 2022	Stated Policies Scenario	Energy	Total	Total energy supply	EJ	World	2020	
2	World Energy Outlook 2022	Stated Policies Scenario	Energy	Total	Total energy supply	EJ	World	2021	
3	World Energy Outlook 2022	Stated Policies Scenario	Energy	Total	Total energy supply	EJ	World	2030	
4	World Energy Outlook 2022	Stated Policies Scenario	Energy	Total	Total energy supply	EJ	World	2050	
...	...	...	...	...	...	...	...	...	
2925	World Energy Outlook 2022	Stated Policies Scenario	CO2 combustion and process	Total	Total final consumption	Mt CO2	Southeast Asia	2021	
2926	World Energy Outlook 2022	Stated Policies Scenario	CO2 combustion and process	Total	Total final consumption	Mt CO2	Southeast Asia	2030	1
2927	World Energy Outlook 2022	Stated Policies Scenario	CO2 combustion and process	Total	Total final consumption	Mt CO2	Southeast Asia	2050	1
2928	World Energy Outlook 2022	Announced Pledges Scenario	CO2 combustion and process	Total	Total final consumption	Mt CO2	Southeast Asia	2030	1
2929	World Energy Outlook 2022	Announced Pledges Scenario	CO2 combustion and process	Total	Total final consumption	Mt CO2	Southeast Asia	2050	

2930 rows × 9 columns

6.2 Value counts

In [5]:
```
df_hist_elec_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10649 entries, 0 to 10648
Data columns (total 12 columns):
 #   Column                                                Non-Null Count
Dtype
---  ------                                                --------------
-----
 0   Entity                                                10649 non-null
object
 1   Code                                                  8347 non-null
object
 2   Year                                                  10649 non-null
int64
 3   Other renewables excluding bioenergy (TWh) (zero filled)  10649 non-null
float64
 4   Electricity from bioenergy (TWh) (zero filled)        10649 non-null
float64
 5   Electricity from solar (TWh)                          8683 non-null
float64
 6   Electricity from wind (TWh)                           8676 non-null
float64
 7   Electricity from hydro (TWh)                          8840 non-null
float64
 8   Electricity from nuclear (TWh)                        8741 non-null
float64
 9   Electricity from oil (TWh)                            6332 non-null
float64
 10  Electricity from gas (TWh)                            6332 non-null
float64
 11  Electricity from coal (TWh)                           6332 non-null
float64
dtypes: float64(9), int64(1), object(2)
memory usage: 998.5+ KB
```

In [6]:
```
df_hist_dem_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5570 entries, 0 to 5569
Data columns (total 4 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Entity                   5570 non-null   object
 1   Code                     5106 non-null   object
 2   Year                     5570 non-null   int64
 3   Electricity demand (TWh)  5570 non-null   float64
dtypes: float64(1), int64(1), object(2)
memory usage: 174.2+ KB
```

In [7]:
```
df_proj_dem_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2930 entries, 0 to 2929
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PUBLICATION  2930 non-null   object
 1   SCENARIO     2930 non-null   object
 2   CATEGORY     2930 non-null   object
 3   PRODUCT      2930 non-null   object
 4   FLOW         2930 non-null   object
 5   UNIT         2930 non-null   object
 6   REGION       2930 non-null   object
 7   YEAR         2930 non-null   int64
 8   VALUE        2930 non-null   float64
dtypes: float64(1), int64(1), object(7)
memory usage: 206.1+ KB
```

7 EDA / Data Cleaning

The columns need more succinct names.

7.1 IEA Region Lists

Unfortunately, for a few critical datasets, the full dataset with individual country data is behind a paywall. The IEA's **World Energy Outlook 2022** Report splits the world into seven regions on Page 499 (https://iea.blob.core.windows.net/assets/830fe099-5530-48f2-a7c1-11f35d510983/WorldEnergyOutlook2022.pdf#page=499):

- North America
- Central & South America
- Europe
- Africa
- Middle East
- Eurasia
- Asia Pacific

Below I'll create lists to group the countries together, along with other groups the report defines.

North America

In [8]:
```
n_america = ['CAN', 'MEX', 'USA']
```

Central & South America

In [9]:
```
cen_s_america = ['ABW', 'AIA', 'ARG', 'ATG', 'BHS', 'BRB', 'BLZ', 'BMU', 'VEN'
                 'BRA', 'VGB', 'CYM', 'CHL', 'COL', 'CRI', 'CUB', 'CUW', 'DMA'
                 'DOM', 'ECU', 'SLV', 'FLK', 'GUF', 'GRD', 'GLP', 'GTM', 'GUY'
                 'HTI', 'HND', 'JAM', 'MTQ', 'MSR', 'NIC', 'PAN', 'PRY', 'PER'
                 'PRI', 'BOL', 'KNA', 'LCA', 'SPM', 'SUR', 'TTO', 'TCA', 'URY'
                 'VCT', 'VIR']
```

Europe

In [10]:
```
european_u = ['AUT', 'BEL', 'BGR', 'HRV', 'CYP', 'CZE', 'DNK', 'EST', 'FIN',
              'FRA', 'DEU', 'GRC', 'HUN', 'IRL', 'ITA', 'LVA', 'LTU', 'LUX',
              'MLT', 'NLD', 'POL', 'PRT', 'ROU', 'SVK', 'SVN', 'ESP', 'SWE']
```

In [11]:
```
europe = european_u + ['ALB', 'AND', 'ANT', 'BLR', 'BIH', 'FRO', 'GIB', 'ISL',
                       'ISR', 'LIE', 'MCO', 'MNE', 'MKD', 'NOR', 'MDA', 'PSE',
                       'SRB', 'SMR', 'CHE', 'TUR', 'UKR', 'GBR', 'GRL', 'VAT',
                       'OWID_KOS', ]
```

Africa

In [12]:
```
n_africa = ['DZA', 'EGY', 'ESH', 'LBY', 'MAR', 'TUN']
```

In [13]:
```
sub_africa = ['AGO', 'BEN', 'BWA', 'BFA', 'BDI', 'CPV', 'CMR', 'CAF', 'TCD',
              'COM', 'CIV', 'COD', 'DJI', 'GNQ', 'ERI', 'ETH', 'GAB', 'GMB',
              'GHA', 'GIN', 'GNB', 'KEN', 'SWZ', 'LSO', 'LBR', 'MDG', 'MWI',
              'MLI', 'MRT', 'MUS', 'MOZ', 'NAM', 'NER', 'NGA', 'COG', 'REU',
              'RWA', 'STP', 'SEN', 'SYC', 'SLE', 'SOM', 'ZAF', 'SSD', 'SDN',
              'TGO', 'UGA', 'TZA', 'ZMB', 'ZWE', 'SHN']
```

In [14]:
```
africa = n_africa + sub_africa
```

Middle East

In [15]:
```
mid_east = ['BHR', 'IRQ', 'IRN', 'JOR', 'KWT', 'LBN', 'OMN', 'QAT', 'SAU',
            'SYR', 'ARE', 'YEM']
```

Eurasia

In [16]:
```
caspian = ['ARM', 'AZE', 'GEO', 'KAZ', 'KGZ', 'TJK', 'TKM', 'UZB']
```

In [17]:
```
eurasia = caspian + ['RUS']
```

Asia Pacific

In [18]:
```
se_asia = ['BRN', 'KHM', 'IDN', 'LAO', 'MYS', 'MMR', 'PHL', 'SGP', 'THA', 'VNM
```

In [19]:
```
asia_pacific = se_asia + ['AFG', 'AUS', 'ASM', 'ATA', 'BGD', 'BTN', 'TWN',
                          'COK', 'PRK', 'FJI', 'FSM', 'GUM', 'HKG', 'IOT',
                          'PYF', 'IND', 'JPN', 'KIR', 'KOR', 'MAC', 'MDV',
                          'MNG', 'NRU', 'NPL', 'NCL', 'NIU', 'NZL', 'PAK',
                          'PNG', 'PLW', 'CHN', 'WSM', 'SLB', 'LKA', 'TKL',
                          'TON', 'TUV', 'TLS', 'VUT']
```

Other Divisions

In [20]:
```
iea_countries = ['AUS', 'AUT', 'BEL', 'CAN', 'CZE', 'DNK', 'EST', 'FIN', 'FRA'
                 'DEU', 'GRC', 'HUN', 'IRL', 'ITA', 'JPN', 'KOR', 'LTU', 'LUX'
                 'MEX', 'MHL', 'NLD', 'NZL', 'NOR', 'POL', 'PRT', 'SVK', 'ESP'
                 'SWE', 'CHE', 'TUR', 'GBR', 'USA']
```

In [21]:
```
oecd = ['AUS', 'AUT', 'BEL', 'CAN', 'CHL', 'COL', 'CRI', 'CZE', 'DNK', 'EST',
        'FIN', 'FRA', 'DEU', 'GRC', 'HUN', 'ISL', 'IRL', 'ISR', 'ITA', 'JPN',
        'KOR', 'LVA', 'LTU', 'LUX', 'MEX', 'NLD', 'NZL', 'NOR', 'POL', 'PRT',
        'SVK', 'SVN', 'ESP', 'SWE', 'CHE', 'TUR', 'GBR', 'USA']
```

In [22]:
```
opec = ['DZA', 'AGO', 'COG', 'GNQ', 'GAB', 'IRN', 'IRQ', 'KWT', 'LBY', 'NGA',
        'SAU', 'ARE', 'VEN']
```

In [23]:
```
adv_eco = oecd + ['BGR', 'HRV', 'CYP', 'MLT', 'ROU']
```

In [24]:
```
world = (n_america + cen_s_america + europe + africa + mid_east + eurasia +
         asia_pacific)

world.sort()
```

In [25]:
```
dev_asia = asia_pacific.copy()
dev_asia.remove('AUS')
dev_asia.remove('JPN')
dev_asia.remove('KOR')
dev_asia.remove('NZL')
```

In [26]:
```
# emg_dev = All other countries not included in the advanced economies regiona
# grouping.
emg_dev = world.copy()
for con in adv_eco:
    emg_dev.remove(con)

emg_dev_hkg = emg_dev + ['HKG']
```

In [27]:
```
l_america = cen_s_america + ['MEX']
```

In [28]:
```
# Non-OECD: All other countries not included in the OECD regional grouping.

non_oecd = world.copy()
for con in oecd:
    non_oecd.remove(con)

non_oecd_hkg = non_oecd + ['HKG']
```

In [29]:
```
# Non-OPEC: All other countries not included in the OPEC regional grouping.

non_opec = world.copy()
for con in opec:
    non_opec.remove(con)

non_opec_hkg = non_opec + ['HKG']
```

7.2 EDA: Energy Production / Consumption

7.2.1 BP: df_hist_elec

In [30]:
```
df_hist_elec = df_hist_elec_raw.copy()

# There's a lot of empty rows in this DataFrame. I need to remove them.

empty = []
for idx in df_hist_elec.index:
    val = np.nansum(df_hist_elec.loc[idx,df_hist_elec.iloc[0:1,3:].columns])
    if (val == 0) | (np.isnan(val)):
        empty.append(idx)

df_hist_elec.drop(index = empty, inplace = True)
df_hist_elec
```

Out[30]:

	Entity	Code	Year	Other renewables excluding bioenergy (TWh) (zero filled)	Electricity from bioenergy (TWh) (zero filled)	Electricity from solar (TWh)	Electricity from wind (TWh)	Electricity from hydro (TWh)	Electri fr nuc (T
0	Afghanistan	AFG	2000	0.0	0.00	0.00	0.0	0.31	
1	Afghanistan	AFG	2001	0.0	0.00	0.00	0.0	0.50	
2	Afghanistan	AFG	2002	0.0	0.00	0.00	0.0	0.56	
3	Afghanistan	AFG	2003	0.0	0.00	0.00	0.0	0.63	
4	Afghanistan	AFG	2004	0.0	0.00	0.00	0.0	0.56	
...	...	...	...	...	...	...	...	...	
10644	Zimbabwe	ZWE	2017	0.0	0.32	0.01	0.0	3.97	
10645	Zimbabwe	ZWE	2018	0.0	0.39	0.02	0.0	5.05	
10646	Zimbabwe	ZWE	2019	0.0	0.38	0.03	0.0	4.17	
10647	Zimbabwe	ZWE	2020	0.0	0.35	0.03	0.0	3.81	
10648	Zimbabwe	ZWE	2021	0.0	0.38	0.04	0.0	4.00	

8441 rows × 12 columns

In [31]:
```
# I want to create a dictionary to reference the country codes used in the
# electricity production data.

country_code_elec = {}
# There are some empty countries that have entity data. Creating a list of the
country_nan_elec = []

for i in df_hist_elec.index:
    # List of rows without a country
    if (type(df_hist_elec.loc[i, 'Code']) != str):
        if df_hist_elec.loc[i, 'Entity'] not in country_nan_elec:
            country_nan_elec.append(df_hist_elec.loc[i, 'Entity'])

    # List of rows with countries
    else:
        if df_hist_elec.loc[i, 'Code'] not in country_code_elec:
            country_code_elec[df_hist_elec.loc[
                i, 'Code']] = df_hist_elec.loc[i, 'Entity']
```

In [32]:
```
# Renaming the columns of the electricity genation DataFrame and sorting.
renaming = {'Entity' : 'Country',
            'Other renewables excluding bioenergy (TWh) (zero filled)' :
            'Other Renewables (TWh)',
            'Electricity from bioenergy (TWh) (zero filled)' :
            'Bioenergy (TWh)',
            'Electricity from solar (TWh)' : 'Solar (TWh)',
            'Electricity from wind (TWh)' : 'Wind (TWh)',
            'Electricity from hydro (TWh)' : 'Hydro (TWh)',
            'Electricity from nuclear (TWh)':'Nuclear (TWh)',
            'Electricity from oil (TWh)' : 'Oil (TWh)',
            'Electricity from gas (TWh)' : 'Gas (TWh)',
            'Electricity from coal (TWh)' : 'Coal (TWh)'}

df_hist_elec.rename(columns = renaming, inplace = True)
df_hist_elec.sort_values(by=['Code', 'Year'],inplace=True)
```

In [33]:
```
# Defining the years available in this data.
print('The electricity generation data includes the years',
      df_hist_elec['Year'].min(), '-', str(df_hist_elec['Year'].max())+'.')

# I'll create a dataframe to place rearranged electricity data into.
df_hist_elec_yr = pd.DataFrame(data = range(df_hist_elec['Year'].min(),
                                            df_hist_elec['Year'].max()+1),
                                            columns = ['Year'])

df_hist_elec_yr.index = df_hist_elec_yr['Year']
df_hist_elec_yr.drop(columns = 'Year', inplace = True)
```

```
The electricity generation data includes the years 1965 - 2022.
```

In [34]:
```
# Need to add how this data defines entire world data, 'OWID_WRL' to the rest
# of the country data.
world_ptot = world + ['OWID_WRL']
world_ptot.sort()

# Transposing the DataFrame.
for code in world_ptot:
    temp = df_hist_elec.loc[df_hist_elec['Code'] == code]
    temp.columns = [code + ' ' + col for col in temp.columns.values]
    temp = temp.iloc[:,2:]
    temp.rename(columns = {temp.columns[0] : 'Year'}, inplace = True)
    temp.index = temp['Year']
    temp = temp.iloc[:,1:]
    for col in temp.columns:
        if temp[col].sum() == 0:
            temp.drop(columns = col, inplace = True)
    df_hist_elec_yr = pd.concat([df_hist_elec_yr, temp], axis=1)

df_hist_elec_yr
```

Out[34]:

Year	ABW Solar (TWh)	ABW Wind (TWh)	ABW Oil (TWh)	AFG Solar (TWh)	AFG Hydro (TWh)	AFG Oil (TWh)	AGO Bioenergy (TWh)	AGO Solar (TWh)	AGO Hydro (TWh)	AGO Oil (TWh)	AGO Gas (TWh)	AL Sol (TW
1965	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1966	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1967	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1968	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1969	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1970	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1971	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1972	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1973	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1974	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1975	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1976	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1977	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1978	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1979	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1980	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1981	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1982	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1983	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1984	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1985	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1986	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1987	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1988	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1989	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1990	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.(
1991	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.(
1992	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.(
1993	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.(
1994	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.(
1995	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.(
1996	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.(
1997	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.(

Year	ABW Solar (TWh)	ABW Wind (TWh)	ABW Oil (TWh)	AFG Solar (TWh)	AFG Hydro (TWh)	AFG Oil (TWh)	AGO Bioenergy (TWh)	AGO Solar (TWh)	AGO Hydro (TWh)	AGO Oil (TWh)	AGO Gas (TWh)	AL Sol (TW
1998	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.(
1999	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.(
2000	0.00	0.00	0.73	0.00	0.31	0.16	0.00	0.00	0.90	0.50	0.00	0.(
2001	0.00	0.00	0.76	0.00	0.50	0.09	0.00	0.00	1.01	0.58	0.00	0.(
2002	0.00	0.00	0.77	0.00	0.56	0.13	0.00	0.00	1.13	0.58	0.00	0.(
2003	0.00	0.00	0.79	0.00	0.63	0.31	0.00	0.00	1.23	0.71	0.00	0.(
2004	0.00	0.00	0.81	0.00	0.56	0.33	0.00	0.00	1.73	0.45	0.00	0.(
2005	0.00	0.00	0.86	0.00	0.59	0.34	0.00	0.00	2.20	0.53	0.00	0.(
2006	0.00	0.00	0.85	0.00	0.64	0.20	0.00	0.00	2.64	0.60	0.00	0.(
2007	0.00	0.00	0.88	0.00	0.75	0.20	0.00	0.00	2.47	0.68	0.00	0.(
2008	0.00	0.00	0.86	0.00	0.54	0.19	0.00	0.00	3.10	0.96	0.00	0.(
2009	0.00	0.03	0.87	0.00	0.78	0.16	0.00	0.00	3.06	1.54	0.00	0.(
2010	0.00	0.11	0.78	0.00	0.75	0.19	0.00	0.01	3.67	1.64	0.00	0.(
2011	0.00	0.11	0.77	0.00	0.60	0.18	0.00	0.01	3.97	1.55	0.00	0.(
2012	0.00	0.14	0.73	0.03	0.71	0.14	0.00	0.01	3.73	2.29	0.00	0.(
2013	0.00	0.15	0.74	0.03	0.86	0.22	0.00	0.01	4.72	3.24	0.00	0.(
2014	0.01	0.15	0.73	0.03	0.97	0.16	1.14	0.02	4.99	3.07	0.00	0.(
2015	0.01	0.17	0.75	0.03	1.00	0.15	1.15	0.02	5.04	3.10	0.00	0.(
2016	0.01	0.13	0.76	0.04	1.02	0.15	1.20	0.02	5.76	3.23	0.00	0.(
2017	0.01	0.13	0.79	0.04	1.05	0.18	0.17	0.02	7.58	0.46	2.44	0.(
2018	0.01	0.14	0.76	0.04	0.93	0.20	0.17	0.02	9.79	0.46	2.40	0.(
2019	0.01	0.14	0.77	0.05	0.84	0.18	0.25	0.02	10.87	0.68	3.58	0.(
2020	0.01	0.14	0.73	0.06	0.62	0.12	0.26	0.02	11.95	0.70	3.67	0.(
2021	0.01	0.14	0.78	0.08	0.62	0.13	0.28	0.02	11.50	0.74	3.89	Na
2022	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

In [35]:
```
# Checking for the number of NaN values.
for row in df_hist_elec_yr.index:
    print(row, np.sum(np.sum(df_hist_elec_yr.loc[row].isna())))
```

```
1965 754
1966 754
1967 754
1968 748
1969 748
1970 748
1971 744
1972 744
1973 744
1974 744
1975 744
1976 744
1977 744
1978 744
1979 744
1980 744
1981 740
1982 740
1983 740
1984 740
1985 625
1986 625
1987 625
1988 625
1989 625
1990 509
1991 509
1992 509
1993 509
1994 509
1995 509
1996 509
1997 509
1998 509
1999 509
2000 9
2001 6
2002 6
2003 5
2004 5
2005 2
2006 2
2007 2
2008 2
2009 2
2010 3
2011 3
2012 16
2013 16
2014 16
2015 16
2016 16
2017 16
2018 16
2019 16
2020 16
```

```
2021 25
2022 810
```

In [36]:
```
# There are only twenty five values in 2021 that are missing.
# I'll see what they are.

na_2021 = []
na_2021_test = df_hist_elec_yr.loc[2021].isna()

for truth in na_2021_test.index:
    if na_2021_test[truth] == True:
        na_2021.append(truth)

df_hist_elec_yr.loc[1990:,na_2021]
```

Out[36]:

Year	ALB Solar (TWh)	ALB Hydro (TWh)	ALB Oil (TWh)	ESH Oil (TWh)	GLP Other Renewables (TWh)	GLP Bioenergy (TWh)	GLP Solar (TWh)	GLP Wind (TWh)	GLP Coal (TWh)	GUF Bioenergy (TWh)	S (T
1990	0.00	2.85	0.45	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1991	0.00	3.52	0.30	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1992	0.00	3.23	0.17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1993	0.00	3.31	0.22	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1994	0.00	3.77	0.17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1995	0.00	4.20	0.27	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1996	0.00	5.73	0.26	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1997	0.00	5.03	0.20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1998	0.00	4.92	0.19	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1999	0.00	5.28	0.14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2000	0.00	4.59	0.18	0.08	0.03	0.39	0.00	0.02	0.83	0.00	
2001	0.00	3.56	0.17	0.08	0.03	0.39	0.00	0.02	0.82	0.00	
2002	0.00	3.51	0.23	0.09	0.03	0.37	0.00	0.03	0.79	0.00	
2003	0.00	4.89	0.11	0.09	0.10	0.37	0.00	0.05	0.79	0.00	
2004	0.00	5.47	0.14	0.09	0.10	0.45	0.00	0.05	0.96	0.00	
2005	0.00	5.37	0.07	0.09	0.10	0.45	0.00	0.05	0.97	0.00	
2006	0.00	5.43	0.09	0.09	0.10	0.48	0.00	0.05	1.01	0.00	
2007	0.00	2.79	0.07	0.09	0.10	0.50	0.00	0.05	1.06	0.00	
2008	0.00	3.80	0.00	0.09	0.10	0.49	0.00	0.05	1.05	0.00	
2009	0.00	5.20	0.00	0.09	0.10	0.49	0.00	0.05	1.04	0.00	
2010	0.00	7.57	0.00	NaN	0.10	0.49	0.02	0.04	1.03	0.01	
2011	0.00	4.13	0.06	NaN	0.10	0.35	0.03	0.05	1.17	0.01	
2012	0.00	4.72	0.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2013	0.00	6.96	0.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2014	0.00	4.72	0.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2015	0.00	5.89	0.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2016	0.00	7.78	0.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2017	0.00	4.52	0.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2018	0.00	8.55	0.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2019	0.02	5.18	0.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2020	0.03	5.28	0.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2021	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

Year	ALB Solar (TWh)	ALB Hydro (TWh)	ALB Oil (TWh)	ESH Oil (TWh)	GLP Other Renewables (TWh)	GLP Bioenergy (TWh)	GLP Solar (TWh)	GLP Wind (TWh)	GLP Coal (TWh)	GUF Bioenergy (TWh)	S (T
2022	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

Since these values are at the end of the data, I have nothing to interpolate. I'll put in some values in the 2021 column to interpolate down each column to that. I based the numbers below on the the trends I observed in the column above it.

I'll also drop the entire 2022 column.

In [37]:
```
df_hist_elec_yr.loc[2021,na_2021] = [0.04,6.24,0,0.11,0.1,0.45,0.13,0.05,1.44,
                                     0.06,0.44,0.66,0.56,0,1.48,0.01,0.09,0.18
                                     0.01,6.64,0.5,0.74,0.02,0.54,2.67]

df_hist_elec_yr.drop(index = 2022, inplace = True)

df_hist_elec_yr.loc[1990:,na_2021] = df_hist_elec_yr.loc[
    1990:,na_2021].interpolate('linear')

df_hist_elec_yr.loc[1990:,na_2021]
```

Out[37]:

Year	ALB Solar (TWh)	ALB Hydro (TWh)	ALB Oil (TWh)	ESH Oil (TWh)	GLP Other Renewables (TWh)	GLP Bioenergy (TWh)	GLP Solar (TWh)	GLP Wind (TWh)	GLP Coal (TWh)	GUF Bioenergy (TWh)
1990	0.00	2.85	0.45	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1991	0.00	3.52	0.30	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1992	0.00	3.23	0.17	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1993	0.00	3.31	0.22	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1994	0.00	3.77	0.17	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1995	0.00	4.20	0.27	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1996	0.00	5.73	0.26	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1997	0.00	5.03	0.20	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1998	0.00	4.92	0.19	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1999	0.00	5.28	0.14	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2000	0.00	4.59	0.18	0.080000	0.03	0.39	0.00	0.02	0.830	0.000
2001	0.00	3.56	0.17	0.080000	0.03	0.39	0.00	0.02	0.820	0.000
2002	0.00	3.51	0.23	0.090000	0.03	0.37	0.00	0.03	0.790	0.000
2003	0.00	4.89	0.11	0.090000	0.10	0.37	0.00	0.05	0.790	0.000
2004	0.00	5.47	0.14	0.090000	0.10	0.45	0.00	0.05	0.960	0.000
2005	0.00	5.37	0.07	0.090000	0.10	0.45	0.00	0.05	0.970	0.000
2006	0.00	5.43	0.09	0.090000	0.10	0.48	0.00	0.05	1.010	0.000
2007	0.00	2.79	0.07	0.090000	0.10	0.50	0.00	0.05	1.060	0.000
2008	0.00	3.80	0.00	0.090000	0.10	0.49	0.00	0.05	1.050	0.000
2009	0.00	5.20	0.00	0.090000	0.10	0.49	0.00	0.05	1.040	0.000
2010	0.00	7.57	0.00	0.091667	0.10	0.49	0.02	0.04	1.030	0.010
2011	0.00	4.13	0.06	0.093333	0.10	0.35	0.03	0.05	1.170	0.010
2012	0.00	4.72	0.00	0.095000	0.10	0.36	0.04	0.05	1.197	0.015
2013	0.00	6.96	0.00	0.096667	0.10	0.37	0.05	0.05	1.224	0.020
2014	0.00	4.72	0.00	0.098333	0.10	0.38	0.06	0.05	1.251	0.025
2015	0.00	5.89	0.00	0.100000	0.10	0.39	0.07	0.05	1.278	0.030
2016	0.00	7.78	0.00	0.101667	0.10	0.40	0.08	0.05	1.305	0.035
2017	0.00	4.52	0.00	0.103333	0.10	0.41	0.09	0.05	1.332	0.040
2018	0.00	8.55	0.00	0.105000	0.10	0.42	0.10	0.05	1.359	0.045
2019	0.02	5.18	0.00	0.106667	0.10	0.43	0.11	0.05	1.386	0.050
2020	0.03	5.28	0.00	0.108333	0.10	0.44	0.12	0.05	1.413	0.055
2021	0.04	6.24	0.00	0.110000	0.10	0.45	0.13	0.05	1.440	0.060

From the year 2000 to 2010, just nine countries are missing data. I'll take a look at what's going on.

In [38]:
```
# Creating a list of indexes

#List of columns that are missing values.
na_2000 = []
# List of columns as the index for which columns have NaN in the year 2000.
na_2000_test = df_hist_elec_yr.loc[2000].isna()

# Creating a list of just the column names with NaNs in the year 2000.
for truth in na_2000_test.index:
    if na_2000_test[truth] == True:
        na_2000.append(truth)

df_hist_elec_yr.loc[2000:,na_2000]
```

Out[38]:

Year	MNE Wind (TWh)	MNE Hydro (TWh)	MNE Coal (TWh)	PSE Solar (TWh)	PSE Oil (TWh)	PSE Gas (TWh)	SSD Solar (TWh)	SSD Oil (TWh)	TLS Oil (TWh)
2000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2001	NaN	NaN	NaN	0.00	0.00	0.03	NaN	NaN	NaN
2002	NaN	NaN	NaN	0.00	0.07	0.07	NaN	NaN	NaN
2003	NaN	NaN	NaN	0.00	0.16	0.16	NaN	NaN	0.06
2004	NaN	NaN	NaN	0.00	0.19	0.19	NaN	NaN	0.07
2005	0.00	1.87	1.00	0.00	0.23	0.23	NaN	NaN	0.07
2006	0.00	1.75	1.20	0.00	0.16	0.16	NaN	NaN	0.07
2007	0.00	1.28	0.86	0.00	0.20	0.20	NaN	NaN	0.09
2008	0.00	1.54	1.29	0.00	0.20	0.20	NaN	NaN	0.11
2009	0.00	2.07	0.69	0.00	0.23	0.23	NaN	NaN	0.13
2010	0.00	2.75	1.27	0.00	0.22	0.22	NaN	NaN	0.14
2011	0.00	1.20	1.45	0.00	0.27	0.27	NaN	NaN	0.14
2012	0.00	1.48	1.37	0.00	0.22	0.22	0.00	0.42	0.13
2013	0.00	2.50	1.44	0.00	0.25	0.25	0.00	0.44	0.29
2014	0.00	1.75	1.42	0.00	0.16	0.16	0.00	0.46	0.35
2015	0.00	1.49	1.51	0.01	0.25	0.22	0.00	0.54	0.35
2016	0.00	1.84	1.30	0.04	0.25	0.22	0.00	0.49	0.41
2017	0.10	1.02	1.36	0.05	0.25	0.22	0.00	0.51	0.43
2018	0.14	2.11	1.55	0.06	0.18	0.16	0.01	0.55	0.45
2019	0.30	1.63	1.50	0.12	0.35	0.30	0.01	0.55	0.52
2020	0.32	1.45	1.62	0.18	0.33	0.29	0.01	0.53	0.50
2021	0.33	1.98	1.50	0.20	0.35	0.31	0.01	0.56	0.53

To fill this in, I used the following strategies, calculating them separately in Microsoft Excel:

- MNE Wind (TWh): Copied zeros on up.
- MNE Hydro (TWh): Took the forward ten year average for the country and walked it backwards.
- MNE Coal (TWh): Took the forward ten year average for the country and walked it backwards.
- PSE: Three zeros
- SSD Solar (TWh): Copied zeros on up.
- SSD Oil (TWh): Between 2012 and 2021, the country added .1 TWh to their grid. So I projected that backwards to the year 2000 at a steady rate.
- TLS Oil (TWh): copied .06 on up.

Below I With these relatively few assumptions, I can keep ten years worth of data.

In [39]:
```
pd.set_option('display.max_rows', 30)
# Setting the values as described above.
df_hist_elec_yr.loc[2000,na_2000] = [0, 1.85, 1.1, 0, 0, 0, 0, 0.3, 0.06]
df_hist_elec_yr.loc[
    2000:,na_2000]=df_hist_elec_yr.loc[2000:,na_2000].interpolate()
df_hist_elec_yr.loc[2000:,na_2000]
```

Out[39]:

Year	MNE Wind (TWh)	MNE Hydro (TWh)	MNE Coal (TWh)	PSE Solar (TWh)	PSE Oil (TWh)	PSE Gas (TWh)	SSD Solar (TWh)	SSD Oil (TWh)	TLS Oil (TWh)
2000	0.00	1.850	1.10	0.00	0.00	0.00	0.00	0.30	0.06
2001	0.00	1.854	1.08	0.00	0.00	0.03	0.00	0.31	0.06
2002	0.00	1.858	1.06	0.00	0.07	0.07	0.00	0.32	0.06
2003	0.00	1.862	1.04	0.00	0.16	0.16	0.00	0.33	0.06
2004	0.00	1.866	1.02	0.00	0.19	0.19	0.00	0.34	0.07
2005	0.00	1.870	1.00	0.00	0.23	0.23	0.00	0.35	0.07
2006	0.00	1.750	1.20	0.00	0.16	0.16	0.00	0.36	0.07
2007	0.00	1.280	0.86	0.00	0.20	0.20	0.00	0.37	0.09
2008	0.00	1.540	1.29	0.00	0.20	0.20	0.00	0.38	0.11
2009	0.00	2.070	0.69	0.00	0.23	0.23	0.00	0.39	0.13
2010	0.00	2.750	1.27	0.00	0.22	0.22	0.00	0.40	0.14
2011	0.00	1.200	1.45	0.00	0.27	0.27	0.00	0.41	0.14
2012	0.00	1.480	1.37	0.00	0.22	0.22	0.00	0.42	0.13
2013	0.00	2.500	1.44	0.00	0.25	0.25	0.00	0.44	0.29
2014	0.00	1.750	1.42	0.00	0.16	0.16	0.00	0.46	0.35
2015	0.00	1.490	1.51	0.01	0.25	0.22	0.00	0.54	0.35
2016	0.00	1.840	1.30	0.04	0.25	0.22	0.00	0.49	0.41
2017	0.10	1.020	1.36	0.05	0.25	0.22	0.00	0.51	0.43
2018	0.14	2.110	1.55	0.06	0.18	0.16	0.01	0.55	0.45
2019	0.30	1.630	1.50	0.12	0.35	0.30	0.01	0.55	0.52
2020	0.32	1.450	1.62	0.18	0.33	0.29	0.01	0.53	0.50
2021	0.33	1.980	1.50	0.20	0.35	0.31	0.01	0.56	0.53

Now I'll drop all the rows with missing values.

In [40]:
```
df_hist_elec_yr = df_hist_elec_yr.loc[2000:]
df_hist_elec_yr
```

Out[40]:

Year	ABW Solar (TWh)	ABW Wind (TWh)	ABW Oil (TWh)	AFG Solar (TWh)	AFG Hydro (TWh)	AFG Oil (TWh)	AGO Bioenergy (TWh)	AGO Solar (TWh)	AGO Hydro (TWh)	AGO Oil (TWh)	AGO Gas (TWh)	AL Sol (TW
2000	0.00	0.00	0.73	0.00	0.31	0.16	0.00	0.00	0.90	0.50	0.00	0.(
2001	0.00	0.00	0.76	0.00	0.50	0.09	0.00	0.00	1.01	0.58	0.00	0.(
2002	0.00	0.00	0.77	0.00	0.56	0.13	0.00	0.00	1.13	0.58	0.00	0.(
2003	0.00	0.00	0.79	0.00	0.63	0.31	0.00	0.00	1.23	0.71	0.00	0.(
2004	0.00	0.00	0.81	0.00	0.56	0.33	0.00	0.00	1.73	0.45	0.00	0.(
2005	0.00	0.00	0.86	0.00	0.59	0.34	0.00	0.00	2.20	0.53	0.00	0.(
2006	0.00	0.00	0.85	0.00	0.64	0.20	0.00	0.00	2.64	0.60	0.00	0.(
2007	0.00	0.00	0.88	0.00	0.75	0.20	0.00	0.00	2.47	0.68	0.00	0.(
2008	0.00	0.00	0.86	0.00	0.54	0.19	0.00	0.00	3.10	0.96	0.00	0.(
2009	0.00	0.03	0.87	0.00	0.78	0.16	0.00	0.00	3.06	1.54	0.00	0.(
2010	0.00	0.11	0.78	0.00	0.75	0.19	0.00	0.01	3.67	1.64	0.00	0.(
2011	0.00	0.11	0.77	0.00	0.60	0.18	0.00	0.01	3.97	1.55	0.00	0.(
2012	0.00	0.14	0.73	0.03	0.71	0.14	0.00	0.01	3.73	2.29	0.00	0.(
2013	0.00	0.15	0.74	0.03	0.86	0.22	0.00	0.01	4.72	3.24	0.00	0.(
2014	0.01	0.15	0.73	0.03	0.97	0.16	1.14	0.02	4.99	3.07	0.00	0.(
2015	0.01	0.17	0.75	0.03	1.00	0.15	1.15	0.02	5.04	3.10	0.00	0.(
2016	0.01	0.13	0.76	0.04	1.02	0.15	1.20	0.02	5.76	3.23	0.00	0.(
2017	0.01	0.13	0.79	0.04	1.05	0.18	0.17	0.02	7.58	0.46	2.44	0.(
2018	0.01	0.14	0.76	0.04	0.93	0.20	0.17	0.02	9.79	0.46	2.40	0.(
2019	0.01	0.14	0.77	0.05	0.84	0.18	0.25	0.02	10.87	0.68	3.58	0.(
2020	0.01	0.14	0.73	0.06	0.62	0.12	0.26	0.02	11.95	0.70	3.67	0.(
2021	0.01	0.14	0.78	0.08	0.62	0.13	0.28	0.02	11.50	0.74	3.89	0.(

Creating DataFrame of the historic electricity generation methods by region as defined by the IEA.

In [41]:
```
# This cell takes the DataFrame and sums the data into the IEA's regions.

df_hist_elec_yr = df_hist_elec_yr.copy()

pd.set_option('display.max_rows', 10)

# The columns all have one of the strings below.
col_type = [' Other Renewables (TWh)', ' Bioenergy (TWh)', ' Solar (TWh)',
            ' Wind (TWh)', ' Hydro (TWh)', ' Nuclear (TWh)', ' Oil (TWh)',
            ' Gas (TWh)', ' Coal (TWh)']

# List for all regions column names
reg_col = []
# List for just Earth's column names.
world_col = []
# For renaming OWID_WRL to Earth
rename = []
# Creating columns for Earth
for c in col_type:
    reg_col.append('OWID_WRL' + c)
    world_col.append('OWID_WRL' + c)
    rename.append('Earth' + c)

# Renaming OWID_WRL to Earth
renaming = dict(zip(world_col, rename))

# Creating all regions column names.
reg = ['Asia Pacific', 'North America', 'Europe', 'Eurasia',
       'Central & South America', 'Middle East', 'Africa', 'USA', 'CHN',
       'European Union', 'JPN', 'RUS', 'IND',
       'SouthEast Asia', 'BRA']

# Creating all regions column names.
for r in reg:
    for c in col_type:
        reg_col.append(r+c)

# Creating the DataFrame with the regions column names.
df_hist_elec_yr_reg = pd.DataFrame(data = 0, columns = reg_col,
                                   index = df_hist_elec_yr.index)

# Since the world columns are already calculated, copying over that data.
df_hist_elec_yr_reg[world_col] = df_hist_elec_yr[world_col]

# Copying over country information
for col in df_hist_elec_yr_reg.columns:
    if col in df_hist_elec_yr.columns:
        df_hist_elec_yr_reg[col] = df_hist_elec_yr[col]

# Many of the variables listed below are defined above. They each consist of
# lists of the countries in that region. Each country defined by their three
# letter ISO.
# This adds the value of each country's emissions into that country's region.
# This variable keeps track of any missing countries
missing_coun = []
# Coun iterates through all the world's countries.
for coun in world:
```

```
    # t is the GHG and total or fuel.
    for t in col_type:
        # The country and the GHG / Total or Fuel
        tst = coun + t
        # Tests for errors.
        r = '?'
        # If the column doesn't exist in the data, move on.
        try:
            col = df_hist_elec_yr[tst].copy()
        except:
            continue
        if coun in asia_pacific:
            r = 'Asia Pacific' + t
        if coun in n_america:
            r = 'North America' + t
        if coun in europe:
            r = 'Europe' + t
        if coun in eurasia:
            r = 'Eurasia' + t
        if coun in cen_s_america:
            r = 'Central & South America' + t
        if coun in mid_east:
            r = 'Middle East' + t
        if coun in africa:
            r = 'Africa' + t
        if coun in european_u:
            r = 'European Union' + t
        if coun in se_asia:
            r = 'SouthEast Asia' + t
        # If the column didn't exist, put it in the missing data list.
        if r == '?':
            print("No data for", coun)
            missing_coun.append(coun)
            continue
        # Otherwise, add that country's data to that country's region column.
        df_hist_elec_yr_reg[r] = df_hist_elec_yr_reg[r] + df_hist_elec_yr[tst]

# Turning the beginning ISO into the spelled out name.
def rename_coun(df):
    for col in df.columns:
        if col[0:3] == 'USA':
            d = {col : 'United States' + col[3:]}
        elif col[0:3] == 'CHN':
            d = {col : 'China' + col[3:]}
        elif col[0:3] == 'JPN':
            d = {col : 'Japan' + col[3:]}
        elif col[0:3] == 'RUS':
            d = {col : 'Russia' + col[3:]}
        elif col[0:3] == 'IND':
            d = {col : 'India' + col[3:]}
        elif col[0:3] == 'BRA':
            d = {col : 'Brazil' + col[3:]}
        else:
            continue
        df.rename(columns = d, inplace = True)
    return df
```

```
# Renaming OWID_WRL to Earth
df_hist_elec_yr_reg.rename(columns = renaming, inplace = True)

# Turning the beginning ISO into the spelled out name.
df_hist_elec_yr_reg = rename_coun(df_hist_elec_yr_reg)
df_hist_elec_yr_reg
```

Out[41]:

Year	Earth Other Renewables (TWh)	Earth Bioenergy (TWh)	Earth Solar (TWh)	Earth Wind (TWh)	Earth Hydro (TWh)	Earth Nuclear (TWh)	Earth Oil (TWh)	Earth Gas (TWh)	Earth Coal (TWh)	A R
2000	52.37	148.41	1.08	31.16	2621.47	2507.43	1173.34	2717.70	5718.94	
2001	52.60	142.90	1.35	38.16	2561.15	2573.71	1157.32	2867.51	5801.76	
2002	54.08	156.22	1.69	52.04	2601.49	2601.89	1135.84	3071.37	6058.69	
2003	56.07	167.59	2.07	63.43	2602.17	2577.71	1154.97	3203.84	6464.10	
2004	57.94	184.57	2.71	85.26	2796.82	2682.73	1140.43	3445.59	6697.23	
...	...	...	...	...	...	...	...	...	...	
2017	86.62	517.97	444.54	1136.41	4054.17	2566.22	872.27	5826.91	9521.74	
2018	89.80	549.25	570.57	1265.29	4174.84	2619.57	799.24	6051.69	9902.09	
2019	91.39	577.97	701.19	1419.53	4220.50	2724.08	738.53	6208.60	9684.42	
2020	94.28	605.08	852.10	1586.92	4340.61	2634.69	710.57	6153.20	9297.78	
2021	95.65	666.28	1040.50	1848.26	4234.35	2739.32	764.52	6337.96	10085.90	

22 rows × 144 columns

7.2.2 BP: df_hist_dem

In [42]:
```
df_hist_dem = df_hist_dem_raw.copy()

# From info above, I saw that there are fewer values in 'Code' than 'Entity'.
# I want to see which ones are missing a code.

df_hist_dem.loc[df_hist_dem['Code'].isna()]['Entity'].unique()
```

Out[42]:
```
array(['Africa', 'Africa (Ember)', 'Asia', 'Asia (Ember)', 'Europe',
       'Europe (Ember)', 'European Union (27)',
       'European Union (27) (Ember)', 'G20 (Ember)', 'G7 (Ember)',
       'High-income countries', 'Latin America and Caribbean (Ember)',
       'Low-income countries', 'Lower-middle-income countries',
       'North America', 'North America (Ember)', 'OECD (Ember)',
       'Oceania', 'Oceania (Ember)', 'South America',
       'Upper-middle-income countries'], dtype=object)
```

In [43]:
```
# This analysis will use the IEA's definitions of the regions, already defined
# above. Therefore, I don't need any of these columns without Codes.
# I will drop those rows.

drop = df_hist_dem.loc[df_hist_dem['Code'].isna()]['Entity'].index
df_hist_dem.drop(drop, inplace = True)
```

In [44]:
```
# The country info I need is in Code, not Entity. I'll drop the column.
df_hist_dem.drop(columns = 'Entity', inplace = True)
df_hist_dem.sort_values(by = ['Code'], inplace = True)
```

In [45]:
```
# Now, I'll transpose the columns so year can be the index.
df_index = list(range(df_hist_dem['Year'].min(),
                      df_hist_dem['Year'].max() + 1))

df_hist_dem_yr = pd.DataFrame(index = df_index)
for code in df_hist_dem['Code'].unique():
    temp = df_hist_dem.loc[df_hist_dem['Code'] == code].iloc[:,1:]
    temp.index = temp['Year'].astype(int)
    temp.drop(columns='Year', inplace=True)
    temp.columns = [code + ' Historic Demand (TWh)']
    df_hist_dem_yr = pd.concat([df_hist_dem_yr, temp], axis=1)

df_hist_dem_yr.index.name = 'Year'
df_hist_dem_yr
```

Out[45]:

Year	ABW Historic Demand (TWh)	AFG Historic Demand (TWh)	AGO Historic Demand (TWh)	ALB Historic Demand (TWh)	ARE Historic Demand (TWh)	ARG Historic Demand (TWh)	ARM Historic Demand (TWh)	ASM Historic Demand (TWh)	ATG Historic Demand (TWh)	His Der (
1990	NaN	NaN	NaN	3.51	NaN	NaN	NaN	NaN	NaN	
1991	NaN	NaN	NaN	2.65	NaN	NaN	NaN	NaN	NaN	
1992	NaN	NaN	NaN	2.89	NaN	NaN	NaN	NaN	NaN	
1993	NaN	NaN	NaN	3.39	NaN	NaN	NaN	NaN	NaN	
1994	NaN	NaN	NaN	3.75	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	
2018	0.91	6.16	12.84	7.64	127.90	149.35	5.96	0.16	0.33	2
2019	0.92	5.98	15.40	7.61	129.64	143.85	6.34	0.16	0.34	2
2020	0.88	5.95	16.60	7.59	126.52	143.40	6.44	0.15	0.33	2
2021	0.93	6.20	16.43	NaN	135.60	149.18	6.69	0.16	0.35	2
2022	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

33 rows × 216 columns

In [46]:
```
# Checking for the number of NaN values.
for row in df_hist_dem_yr.index:
    print(row, np.sum(np.sum(df_hist_dem_yr.loc[row].isna())))
```

```
1990 181
1991 181
1992 181
1993 181
1994 181
1995 181
1996 181
1997 181
1998 181
1999 181
2000 3
2001 3
2002 3
2003 2
2004 2
2005 1
2006 1
2007 1
2008 1
2009 1
2010 1
2011 1
2012 0
2013 0
2014 0
2015 0
2016 0
2017 0
2018 0
2019 0
2020 0
2021 3
2022 189
```

In [47]:
```
# There are only three values in 2021 that are missing. I'll see what they are

na_2021 = []
na_2021_test = df_hist_dem_yr.loc[2021].isna()

for truth in na_2021_test.index:
    if na_2021_test[truth] == True:
        na_2021.append(truth)

df_hist_dem_yr.loc[1990:,na_2021]
```

Out[47]:

Year	ALB Historic Demand (TWh)	ISL Historic Demand (TWh)	OWID_KOS Historic Demand (TWh)
1990	3.51	4.51	NaN
1991	2.65	4.49	NaN
1992	2.89	4.55	NaN
1993	3.39	4.73	NaN
1994	3.75	4.77	NaN
...	...	...	...
2018	7.64	19.82	6.06
2019	7.61	19.49	6.37
2020	7.59	19.13	6.38
2021	NaN	NaN	NaN
2022	NaN	NaN	NaN

33 rows × 3 columns

Since these values are at the end of the data, I have nothing to interpolate. I'll put in some random values that make sense with the rest of it.

To get these three numbers, I took the five year average. Also, I checked. These numbers are not a summation of the produced electricity.

I'll also drop the entire 2022 row.

In [48]:
```
df_hist_dem_yr.loc[2021,na_2021] = [7.6, 19.25, 6.11]

df_hist_dem_yr.drop(index = 2022, inplace = True)

df_hist_dem_yr.loc[2021,na_2021]
```

Out[48]:
```
ALB Historic Demand (TWh)          7.60
ISL Historic Demand (TWh)         19.25
OWID_KOS Historic Demand (TWh)     6.11
Name: 2021, dtype: float64
```

From the year 2000 to 2010, just three countries are missing data. I'll take a look at what's going on.

In [49]:
```
pd.set_option('display.max_rows',30)
na_2000 = []
na_2000_test = df_hist_dem_yr.loc[2000].isna()

for truth in na_2000_test.index:
    if na_2000_test[truth] == True:
        na_2000.append(truth)

df_hist_dem_yr.loc[2000:,na_2000]
```

Out[49]:

Year	MNE Historic Demand (TWh)	SSD Historic Demand (TWh)	TLS Historic Demand (TWh)
2000	NaN	NaN	NaN
2001	NaN	NaN	NaN
2002	NaN	NaN	NaN
2003	NaN	NaN	0.06
2004	NaN	NaN	0.07
2005	4.66	NaN	0.07
2006	4.82	NaN	0.07
2007	4.73	NaN	0.09
2008	4.61	NaN	0.11
2009	3.81	NaN	0.13
2010	4.02	NaN	0.14
2011	4.21	NaN	0.14
2012	4.06	0.42	0.13
2013	3.60	0.44	0.29
2014	3.43	0.46	0.35
2015	3.52	0.54	0.35
2016	3.44	0.49	0.41
2017	3.60	0.51	0.43
2018	3.60	0.56	0.45
2019	3.68	0.56	0.52
2020	3.47	0.54	0.50
2021	3.12	0.57	0.53

To fill this in, I used the following strategies:

- MNE Demand (TWh): Five year average for the next five years moving backwards.

- SSD Demand (TWh): Copied the data from electricity generation.
- TLS Demand (TWh): Copied the data from electricity generation.

With these relatively few assumptions, I can keep ten years worth

In [50]:
```
# Setting the values as described above.
df_hist_dem_yr.loc[2000,na_2000] = [4.62, 0.3, 0.06]

df_hist_dem_yr.loc[
    2000:,na_2000]=df_hist_dem_yr.loc[2000:,na_2000].interpolate()
df_hist_dem_yr.loc[2000:,na_2000]

df_hist_dem_yr.loc[2000:,na_2000]
```

Out[50]:

Year	MNE Historic Demand (TWh)	SSD Historic Demand (TWh)	TLS Historic Demand (TWh)
2000	4.620	0.30	0.06
2001	4.628	0.31	0.06
2002	4.636	0.32	0.06
2003	4.644	0.33	0.06
2004	4.652	0.34	0.07
2005	4.660	0.35	0.07
2006	4.820	0.36	0.07
2007	4.730	0.37	0.09
2008	4.610	0.38	0.11
2009	3.810	0.39	0.13
2010	4.020	0.40	0.14
2011	4.210	0.41	0.14
2012	4.060	0.42	0.13
2013	3.600	0.44	0.29
2014	3.430	0.46	0.35
2015	3.520	0.54	0.35
2016	3.440	0.49	0.41
2017	3.600	0.51	0.43
2018	3.600	0.56	0.45
2019	3.680	0.56	0.52
2020	3.470	0.54	0.50
2021	3.120	0.57	0.53

Now I'll drop all the rows with missing values.

In [51]:
```
df_hist_dem_yr = df_hist_dem_yr.loc[2000:]
df_hist_dem_yr
```

Out[51]:

Year	ABW Historic Demand (TWh)	AFG Historic Demand (TWh)	AGO Historic Demand (TWh)	ALB Historic Demand (TWh)	ARE Historic Demand (TWh)	ARG Historic Demand (TWh)	ARM Historic Demand (TWh)	ASM Historic Demand (TWh)	ATG Historic Demand (TWh)	His Den (T
2000	0.73	0.57	1.40	5.77	37.54	86.47	5.21	0.16	0.14	19
2001	0.76	0.69	1.59	5.48	40.58	88.25	5.20	0.16	0.16	2
2002	0.77	0.79	1.71	5.85	44.04	86.99	4.85	0.17	0.18	2
2003	0.79	1.04	1.94	5.92	46.48	93.17	4.93	0.18	0.20	2
2004	0.81	0.99	2.18	6.09	49.27	99.16	5.15	0.18	0.21	2
2005	0.86	1.03	2.73	5.81	57.05	104.74	5.16	0.18	0.23	2
2006	0.85	1.27	3.24	6.13	62.76	113.76	5.22	0.18	0.24	2
2007	0.88	1.56	3.15	5.69	74.03	118.45	5.55	0.18	0.26	2
2008	0.86	1.48	4.06	6.23	77.06	120.23	5.45	0.19	0.27	2
2009	0.90	2.09	4.60	6.60	82.91	122.46	5.32	0.18	0.31	2
2010	0.89	2.51	5.32	6.62	88.33	128.24	5.37	0.16	0.32	2
2011	0.88	3.03	5.53	7.45	93.20	132.42	5.89	0.16	0.32	2
2012	0.87	3.95	6.03	7.26	99.82	136.30	6.01	0.16	0.31	2
2013	0.89	4.73	7.97	9.28	103.39	140.11	6.14	0.16	0.31	2
2014	0.89	4.87	9.22	7.79	109.54	141.67	6.22	0.16	0.32	2
2015	0.93	4.96	9.31	7.29	119.75	146.75	6.13	0.16	0.33	2
2016	0.90	5.54	10.21	7.74	122.48	149.16	6.00	0.17	0.32	2
2017	0.93	5.88	10.67	7.43	126.44	149.03	6.24	0.17	0.33	2
2018	0.91	6.16	12.84	7.64	127.90	149.35	5.96	0.16	0.33	2
2019	0.92	5.98	15.40	7.61	129.64	143.85	6.34	0.16	0.34	2
2020	0.88	5.95	16.60	7.59	126.52	143.40	6.44	0.15	0.33	2
2021	0.93	6.20	16.43	7.60	135.60	149.18	6.69	0.16	0.35	2

In [52]:
```
# This cell takes the DataFrame and sums the data into the IEA's regions.

df_hist_dem_yr = df_hist_dem_yr.copy()

pd.set_option('display.max_rows', 10)

# The columns will all have the string below.
col_type = [' Historic Demand (TWh)']

# List for all regions column names
reg_col = []
# List for just Earth's column names.
world_col = []
# For renaming OWID_WRL to Earth
rename = []
# Creating columns for Earth
for c in col_type:
    reg_col.append('OWID_WRL' + c)
    world_col.append('OWID_WRL' + c)
    rename.append('Earth' + c)
# Renaming OWID_WRL to Earth
renaming = dict(zip(world_col, rename))

# Creating all regions column names. (reg defined in an earlier cell)
for r in reg:
    for c in col_type:
        reg_col.append(r+c)

# Creating the DataFrame with the regions column names.
df_hist_dem_yr_reg = pd.DataFrame(data = 0, columns = reg_col,
                                  index = df_hist_dem_yr.index)

# Since the world columns are already calculated, copying over that data.
df_hist_dem_yr_reg[world_col] = df_hist_dem_yr[world_col]

# Copying over country information
for col in df_hist_dem_yr_reg.columns:
    if col in df_hist_dem_yr.columns:
        df_hist_dem_yr_reg[col] = df_hist_dem_yr[col]


# Many of the variables listed below are defined above. They each consist of
# lists of the countries in that region. Each country defined by their three
# letter ISO.
# This adds the value of each country's emissions into that country's region.
# This variable keeps track of any missing countries
missing_coun = []
# Coun iterates through all the world's countries.
for coun in world:
    # t is the GHG and total or fuel.
    for t in col_type:
        # The country and the GHG / Total or Fuel
        tst = coun + t
        # Tests for errors.
        r = '?'
        # If the column doesn't exist in the data, move on.
        try:
```

```
            col = df_hist_dem_yr[tst].copy()
        except:
            continue
        if coun in asia_pacific:
            r = 'Asia Pacific' + t
        if coun in n_america:
            r = 'North America' + t
        if coun in europe:
            r = 'Europe' + t
        if coun in eurasia:
            r = 'Eurasia' + t
        if coun in cen_s_america:
            r = 'Central & South America' + t
        if coun in mid_east:
            r = 'Middle East' + t
        if coun in africa:
            r = 'Africa' + t
        if coun in european_u:
            r = 'European Union' + t
        if coun in se_asia:
            r = 'SouthEast Asia' + t
        # If the column didn't exist, put it in the missing data list.
        if r == '?':
            print("No data for", coun)
            missing_coun.append(coun)
            continue
        # Otherwise, add that country's data to that country's region column.
        df_hist_dem_yr_reg[r] = df_hist_dem_yr_reg[r] + df_hist_dem_yr[tst]

# Renaming OWID_WRL to Earth
df_hist_dem_yr_reg.rename(columns=renaming,inplace = True)

# Turning the beginning ISO into the spelled out name.
df_hist_dem_yr_reg = rename_coun(df_hist_dem_yr_reg)
df_hist_dem_yr_reg
```

Out[52]:

Year	Earth Historic Demand (TWh)	Asia Pacific Historic Demand (TWh)	North America Historic Demand (TWh)	Europe Historic Demand (TWh)	Eurasia Historic Demand (TWh)	Central & South America Historic Demand (TWh)	Middle East Historic Demand (TWh)	Africa Historic Demand (TWh)	United States Historic Demand (TWh)	H De
2000	14971.90	3753.73	4583.44	1014.490	987.19	778.62	430.47	423.01	3835.86	1:
2001	15196.46	3922.35	4500.90	1022.648	996.68	768.05	455.59	438.41	3749.60	1-
2002	15733.31	4176.54	4632.77	1031.256	1014.12	792.29	487.66	463.28	3865.21	1(
2003	16291.95	4483.46	4643.67	1048.504	1038.29	831.39	517.99	486.39	3875.36	1:
2004	17093.28	4921.23	4754.45	1071.732	1065.23	877.23	549.85	515.69	3963.26	2
...	...	...	...	...	...	...	...	...	...	
2017	25026.85	10445.66	5000.66	1195.350	1271.50	1271.14	1109.36	805.64	4108.62	6:
2018	26022.34	11158.56	5166.96	1208.640	1288.76	1291.25	1113.83	820.54	4246.01	7
2019	26366.21	11517.82	5097.23	1197.710	1303.09	1293.64	1149.14	827.63	4197.42	7-
2020	26275.23	11708.52	4986.52	1178.120	1295.58	1291.84	1142.57	806.57	4090.49	7
2021	27812.74	12667.08	5118.78	1232.560	1364.76	1362.57	1211.87	839.32	4191.53	8-

22 rows × 16 columns

7.2.3 IEA: df_proj_dem

In [53]:
```
# Renaming Columns
df_proj_dem = df_proj_dem_raw.copy()
renaming = {'PUBLICATION' : 'Publication',
            'SCENARIO' : 'Scenario',
            'CATEGORY' : 'Category',
            'PRODUCT' : 'Product',
            'FLOW' : 'Flow',
            'UNIT' : 'Unit',
            'REGION' : 'Region',
            'YEAR' : 'Year',
            'VALUE' : 'Value'}
df_proj_dem.rename(columns = renaming, inplace = True)
```

In [54]:
```
# I don't know what the data marked "Co2" is in this category, so I'll drop it
df_proj_dem = df_proj_dem.loc[
    df_proj_dem['Category'] == 'Energy']

# Each cell of Publication, and now Category, has the same value, I'll delete
# both columns.
df_proj_dem.drop(columns = ['Publication', 'Category'], inplace = True)

# I'm intrigued by the data in the Scenario column. There are three categories
df_proj_dem['Scenario'].value_counts()
```

Out[54]:
```
Stated Policies Scenario              1796
Announced Pledges Scenario             768
Net Zero Emissions by 2050 Scenario     30
Name: Scenario, dtype: int64
```

In [55]:
```
# The Announced Pledges Scenario could hold some interesting data, as well as
# the Net Zero Emissions by 2050 Scenario. Putting this data into the same
# Dataframe as the information I need will not be possible, so I'll split it
# into three different scenarios.

df_proj_dem_stat = df_proj_dem.loc[
    df_proj_dem['Scenario'] == 'Stated Policies Scenario'].copy()
df_proj_dem_stat.drop(columns = ['Scenario'], inplace = True)

df_proj_dem_ann = df_proj_dem.loc[
    df_proj_dem['Scenario'] == 'Announced Pledges Scenario'].copy()
df_proj_dem_ann.drop(columns = ['Scenario'], inplace = True)

df_proj_zero = df_proj_dem.loc[
    df_proj_dem[
        'Scenario'] == 'Net Zero Emissions by 2050 Scenario'].copy()
df_proj_zero.drop(columns = ['Scenario'], inplace = True)
```

In [56]:
```
# Now I want to see what the Product column does.
df_proj_dem_ann['Product'].value_counts()
```

Out[56]:
```
Total                   252
Natural gas              86
Coal                     84
Renewables               64
Hydrogen                 64
                       ...
Total oil                 2
Hydrogen based fuels      2
Biofuels                  2
Total liquids             2
Diesel                    2
Name: Product, Length: 38, dtype: int64
```

I already have a reliable source for a breakdown of different region's energy production. I was primarily interested in this DataFrame because it shows what the increase in demand will be. The future of energy product is the variable I will adjust in this project. Therefore, I think I only

need the information in the Total Energy needs. I'll drop the rest. The net zero pledges don't

In [57]:
```
df_proj_dem_stat = df_proj_dem_stat.loc[df_proj_dem_stat['Product'] == 'Total'
df_proj_dem_stat.drop(columns = 'Product', inplace = True)

df_proj_dem_ann = df_proj_dem_ann.loc[df_proj_dem_ann['Product'] == 'Total']
df_proj_dem_ann.drop(columns = 'Product', inplace = True)

# Now I'll check the "Flow" column
df_proj_dem_stat['Flow'].value_counts()
```

Out[57]:
```
Total energy supply        80
Electricity generation     80
Total final consumption    80
Industry                   80
Transport                  80
Buildings                  80
Refining capacity          45
Refinery runs              45
Name: Flow, dtype: int64
```

Again, I really only wanted the Total energy supply information from this DataFrame. I looked up each definition of these terms in the report, and learned that only one of them is relevant to this project, defined below.

Industry (https://iea.blob.core.windows.net/assets/830fe099-5530-48f2-a7c1-11f35d510983/WorldEnergyOutlook2022.pdf#page=499): The sector includes fuel used within the manufacturing and construction industries. Key industry branches include iron and steel, chemical and petrochemical, cement, aluminium, and pulp and paper. Use by industries for the transformation of energy into another form or for the production of fuels is excluded and reported separately under other energy sector. There is an exception for fuel transformation in blast furnaces and coke ovens, which are reported within iron and steel. Consumption of fuels for the transport of goods is reported as part of the transport sector, while consumption by off-road vehicles is reported under industry.

Total energy supply (TES) (https://iea.blob.core.windows.net/assets/830fe099-5530-48f2-a7c1-11f35d510983/WorldEnergyOutlook2022.pdf#page=497): Represents domestic demand only and is broken down into electricity and heat generation, other energy sector and total final consumption.

Electricity generation (https://iea.blob.core.windows.net/assets/830fe099-5530-48f2-a7c1-11f35d510983/WorldEnergyOutlook2022.pdf#page=490): Defined as the total amount of electricity generated by power only or combined heat and power plants including generation required for own use. This is also referred to as gross generation.

Buildings (https://iea.blob.core.windows.net/assets/830fe099-5530-48f2-a7c1-11f35d510983/WorldEnergyOutlook2022.pdf#page=488): The buildings sector includes energy used in residential and services buildings. Services buildings include commercial and institutional buildings and other non-specified buildings. Building energy use includes space heating and cooling, water heating, lighting, appliances and cooking equipment.

Total final consumption (TFC) (https://iea.blob.core.windows.net/assets/830fe099-5530-48f2-a7c1-11f35d510983/WorldEnergyOutlook2022.pdf#page=497): Is the sum of consumption by the various end-use sectors. TFC is broken down into energy demand in the following sectors: industry (including manufacturing, mining, chemicals production, blast furnaces and coke ovens), transport, buildings (including residential and services) and other (including agriculture and other non-energy use). It excludes international marine and aviation bunkers, except at world level where it is included in the transport sector.

Transport (https://iea.blob.core.windows.net/assets/830fe099-5530-48f2-a7c1-11f35d510983/WorldEnergyOutlook2022.pdf#page=498): Fuels and electricity used in the transport of goods or people within the national territory irrespective of the economic sector within which the activity occurs. This includes fuel and electricity delivered to vehicles using public roads or for use in rail vehicles; fuel delivered to vessels for domestic navigation; fuel delivered to aircraft for domestic aviation; and energy consumed in the delivery of fuels through

In [58]:
```
# I want to see what these numbers say for the "World" region.
df_proj_dem_stat.loc[(df_proj_dem_stat['Region'] == 'World') &
                     (df_proj_dem_stat['Year'] == 2021)]
```

Out[58]:

	Flow	Unit	Region	Year	Value
2	Total energy supply	EJ	World	2021	624.164
698	Refining capacity	Million barrels per day	World	2021	101.200
703	Refinery runs	Million barrels per day	World	2021	77.900
1157	Electricity generation	TWh	World	2021	28333.900
1941	Total final consumption	EJ	World	2021	439.103
2053	Industry	EJ	World	2021	166.738
2165	Transport	EJ	World	2021	113.433
2277	Buildings	EJ	World	2021	132.436

(I originally had a different version of this file from the IEA. I noticed this egregious error in the units that was corrected in the version I was able to download when I finished this initial review. I'll leave this here because it makes me smile that I found this error, and it was eventually confirmed by the IEA changing the data.)

Wow. These units don't make a lick of sense.

$$28333.9\ TWh \times \frac{3.6PJ}{TwH} = 102002.04\ PJ$$

If these numbers are accurate, that means that in 2021, the world generated more electricity than the total final consumption. Since electricity generation is one of the sectors added to Total Final consumption, that doesn't make a lot of sense.

I opened the report to see if I could confirm what was going on. Fortunately, I was able confirm that this was a mere typo. Table 5.1 on Page 239 (https://iea.blob.core.windows.net/assets/830fe099-5530-48f2-a7c1-

11f35d510983/WorldEnergyOutlook2022.pdf#page=239) of the report shared these same numbers, but confirmed they are not in petajoules, but in exajoules, which equal a thousand petajoules.

$$28333.9\ TWh \times \frac{.0036 EJ}{TwH} = 102\ EJ$$

In [59]:
```
# Fixing the units as described above.
for i in df_proj_dem_stat.index:
    if df_proj_dem_stat.loc[i,'Unit'] == 'PJ':
        df_proj_dem_stat.loc[i,'Unit'] = 'EJ'

for i in df_proj_dem_ann.index:
    if df_proj_dem_ann.loc[i,'Unit'] == 'PJ':
        df_proj_dem_ann.loc[i,'Unit'] = 'EJ'

for i in df_proj_zero.index:
    if df_proj_zero.loc[i,'Unit'] == 'PJ':
        df_proj_zero.loc[i,'Unit'] = 'EJ'

df_proj_dem_stat.loc[(df_proj_dem_stat['Region'] == 'World') &
                 (df_proj_dem_stat['Year'] == 2021)]
```

Out[59]:

	Flow	Unit	Region	Year	Value
2	Total energy supply	EJ	World	2021	624.164
698	Refining capacity	Million barrels per day	World	2021	101.200
703	Refinery runs	Million barrels per day	World	2021	77.900
1157	Electricity generation	TWh	World	2021	28333.900
1941	Total final consumption	EJ	World	2021	439.103
2053	Industry	EJ	World	2021	166.738
2165	Transport	EJ	World	2021	113.433
2277	Buildings	EJ	World	2021	132.436

I'm still not sure which of these values are relevant for this analysis. I'll compare the above numbers to the electricity generation database from BP and see how they compare. Fortunatley, the BP database has a value for the whole world in 2021, split across different electricity genertaion methods. If I add them up, I'll have a number to compare.

In [60]:
```
# Getting the Total Electricity Generation from this IEA data for 2021
IEA_2021 = df_proj_dem_stat.loc[(df_proj_dem_stat['Region'] == 'World') &
                                (df_proj_dem_stat['Year'] == 2021) &
                                (df_proj_dem_stat['Unit'] == 'TWh')].iloc[:,4]
IEA_2021 = float(IEA_2021)
print(IEA_2021)

# Testing the total generation in the BP data
electricity_2021 = df_hist_elec.loc[(df_hist_elec['Country'] == 'World') &
                   (df_hist_elec['Year'] == 2021)].copy()
Twh = electricity_2021.iloc[:,3:].values
BP_2021 = round(np.sum(Twh),1)
percent = round((IEA_2021/BP_2021-1)*100,2)
print('IEA Global Electricity Demand 2021:   ', IEA_2021)
print('BP Global Electricity Generation 2021:', BP_2021)
print('Percentage difference:                ', str(percent)+'%')
```

```
28333.9
IEA Global Electricity Demand 2021:    28333.9
BP Global Electricity Generation 2021: 27812.7
Percentage difference:                 1.87%
```

I'd say a 1.9% difference means I'm on the right track. I'll use the electricity generation values from the IEA database.

In [61]:
```
# I'll now single out Electricity generation in Flow, and drop the column.

df_proj_dem_stat = df_proj_dem_stat.loc[
    df_proj_dem_stat['Flow'] == 'Electricity generation']
df_proj_dem_stat.drop(columns = 'Flow', inplace = True)

df_proj_dem_ann = df_proj_dem_ann.loc[
    df_proj_dem_ann['Flow'] == 'Electricity generation']
df_proj_dem_ann.drop(columns = 'Flow', inplace = True)
```

In [62]:
```
df_proj_dem_stat
```

Out[62]:

	Unit	Region	Year	Value
1155	TWh	World	2010	21538.90
1156	TWh	World	2020	26707.70
1157	TWh	World	2021	28333.90
1158	TWh	World	2030	34833.60
1159	TWh	World	2050	49844.90
...	...	...	...	...
1260	TWh	Southeast Asia	2010	684.92
1261	TWh	Southeast Asia	2020	1116.10
1262	TWh	Southeast Asia	2021	1164.42
1263	TWh	Southeast Asia	2030	1704.27
1264	TWh	Southeast Asia	2050	3142.94

80 rows × 4 columns

In [63]:
```
# Transpose the columns so year is now the index.
def proj_transpose(df):
    year_index = list(df['Year'].unique())
    newdf = pd.DataFrame(index = year_index)
    for row in df.index:
        col = df.loc[row,'Region']+' Projected Demand ('+df.loc[row,'Unit']+')
        newdf.loc[df.loc[row,'Year'],col] = df.loc[row,'Value']
    return newdf

df_proj_dem_stat_yr = proj_transpose(df_proj_dem_stat)
df_proj_dem_ann_yr = proj_transpose(df_proj_dem_ann)
df_proj_dem_stat_yr.insert(0, 'Year', df_proj_dem_stat_yr.index)
df_proj_dem_ann_yr.insert(0, 'Year', df_proj_dem_ann_yr.index)

#Rearranging
df_proj_dem_stat_yr = df_proj_dem_stat_yr.iloc[:,
                                    [0,1,12,2,6,10,4,9,8,3,13,7,15,11,14,16,5]]
df_proj_dem_ann_yr = df_proj_dem_ann_yr.iloc[:,
                                    [0,1,12,2,6,10,4,9,8,3,13,7,15,11,14,16,5]]

df_proj_dem_stat_yr
```

Out[63]:

	Year	World Projected Demand (TWh)	Asia Pacific Projected Demand (TWh)	North America Projected Demand (TWh)	Europe Projected Demand (TWh)	Eurasia Projected Demand (TWh)	Central and South America Projected Demand (TWh)	Middle East Projected Demand (TWh)	Afric Projecte Deman (TWh
2010	2010	21538.9	8287.77	5232.84	4120.40	1251.26	1130.13	829.42	687.1
2020	2020	26707.7	12866.00	5205.30	3955.91	1366.89	1275.88	1202.75	835.0
2021	2021	28333.9	13907.90	5356.51	4181.82	1455.00	1331.03	1232.84	868.7
2030	2030	34833.6	18370.70	5771.40	4691.24	1539.56	1605.23	1651.34	1204.1
2050	2050	49844.9	26573.20	7815.80	5703.31	1937.40	2591.74	2886.16	2337.2

8 Export

In [64]:
```
df_hist_elec_yr_reg.to_csv(
    "Data/Export/df_hist_elec_yr_reg.csv",
    encoding = 'ANSI')
```

In [65]:
```
df_hist_dem_yr_reg.to_csv(
    "Data/Export/df_hist_dem_yr_reg.csv",
    encoding = 'ANSI')
```

In [66]:
```
df_proj_dem_stat_yr.to_csv(
    "Data/Export/df_proj_dem_stat_yr.csv",
    encoding = 'ANSI')
```

See Notebook.ipynb to see what I do with this data.