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 Average Global Temperature Stats

- **IPCC Working Group 1 Assessment Report 6**
  - **Chapter 2 | Changing State of the Climate System**: Report (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_Chapter02.pdf)
    - Figure 2.11 | Annual average global land-ocean surface temperature from 1850–2020 plus or minus the average from 1850–1900 in °C:
      - IPCC Report, Page 316 (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_Chapte Figure (https://www.ipcc.ch/report/ar6/wg1/figures/chapter-2/figure-2-11), References, Page 19 (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_Chapte Dataset (https://dap.ceda.ac.uk/badc/ar6_wg1/data/ch_02/ch2_fig11/v20211207/Figure lower_panel.csv) (Four datasets. The IPCC averages all four together to create Figure 2.11)

Unexplored:

- Annual average global land-ocean surface temperature from 1880–2022 compared to the long-term average from 1951–1980. NASA (https://climate.nasa.gov/vital-signs/global-temperature/), Dataset (https://data.giss.nasa.gov/gistemp/graphs/graph_data/Global_Mean_Estimates_based_on_
- A brilliant graphic from Bloomberg from back in 1995. Possibly update it? Project it towards the future? Bloomberg (https://www.bloomberg.com/graphics/2015-whats-warming-the-

world/) (Multiple datasets and links in this article)

3.2 Atmospheric Greenhouse Gas Concentrations

- **IPCC Working Group 1 Assessment Report 6**
  - **Chapter 7 | The Earth's Energy Budget, Climate Feedbacks and Climate Sensitivity**: Report (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_Chapter07.pdf)
    - 7.SM Chapter 7: The Earth's energy budget, climate feedbacks, 3 and climate sensitivity - Supplementary Material: Report (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_FGD_Chap
      - Table 7.SM.1 | Simplified expressions to compute radiative forcing (RF) from concentrations of greenhouse gases: Report, Page 3 (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_FGD_C
      - Table 7.SM.7 | Metrics for greenhouse gases: Report, Page 24 (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_FGD_C
        - CSV of table on Github (https://github.com/IPCC-WG1/Chapter-7/blob/main/data_output/7sm/metrics_supplement_cleaned.csv) The author's note in the project's GitHub (https://github.com/IPCC-WG1/Chapter-7/) that the published table has two typos, corrected in this linked table.
  - **Annex III | Tables of Historical and Projected Well-mixed Greenhouse Gas Mixing Ratios and Effective Radiative Forcing of All Climate Forcers**: Report (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_AnnexIII.pdf)
    - IPCC 2021 WG1 AR6 Annex III Extended Data depository: IPCC (https://zenodo.org/record/5705391#.ZDHn1HbMK3A)
    - Table AIII.1 | Historical abundances of all GHG: IPCC (https://zenodo.org/record/5705391/files/table_A3.1_historical_abundances.csv)
    - Table AIII.2 | Future abundances of GHG for nine SSP scenarios (2020–2500): Meinshausen (https://greenhousegases.science.unimelb.edu.au/img/SUPPLEMENT_DataTables_
    - Table AIII.3 | Effective radiative forcing (W m^{-2}) time series of all climate forcers from 1750–2019: IPCC (https://zenodo.org/record/5705391/files/table_A3.3_historical_ERF_1750-2019_best_estimate.csv)
      - Minor GHG breakdown: IPCC (https://zenodo.org/record/5705391/files/table_A3.3_historical_ERF_1750-2019_minorGHG_breakdown.csv)
    - Table AIII.4 | Effective radiative forcing (W m^{-2}) time series of all climate forcers for nine SSP scenarios (2020–2500): IPCC (https://zenodo.org/record/5705391#.ZDHn1HbMK3A)
      - (Currently, this just links back to the full depository. I'll need to make decisions later about what SSP scenarios to compare my projections to)

Unexplored: Level-Up: Possibly add historic data to make up for the missing years of 2020-2022, the pandemic years.

- NOAA's historic atmospheric concentration data on CO2 from 1979–2022: NOAA (https://gml.noaa.gov/webdata/ccgg/trends/co2/co2_annmean_gl.csv)
- NOAA's historic atmospheric concentration data on CH4 from 1984–2022: NOAA (https://gml.noaa.gov/webdata/ccgg/trends/ch4/ch4_annmean_gl.csv)
- NOAA's historic atmospheric concentration data on N2O from 2001–2022: NOAA (https://gml.noaa.gov/webdata/ccgg/trends/n2o/n2o_annmean_gl.csv)
- NOAA's historic atmospheric concentration data on SF6 from 1998–2022: NOAA (https://gml.noaa.gov/webdata/ccgg/trends/sf6/sf6_annmean_gl.csv)
- NOAA's historic atmospheric concentration data of all GHG from 1979–2021: NOAA (https://gml.noaa.gov/aggi/NOAA_MoleFractions_2022.csv)
- NOAA's historic Annual Greenhouse Gas Index (AGGI) from 1979–2021: NOAA (https://gml.noaa.gov/aggi/AGGI_Table.csv)
- NOAA's formulas to convert ppm to Effective Radiative Forcing: NOAA (https://gml.noaa.gov/aggi/aggi.html)
  - I'll only use this in conjunction with other NOAA calculations. Otherwise, I'll use the AR6 formula above.

3.3 Greenhouse Gas Emissions

- Historic emissions data by country by sector: PRIMAP-hist national historical emissions time series (https://zenodo.org/record/7727475#.ZC73QXbMK3A)
  - Key to emissions data: National Greenhouse Gas Inventory, Page 6 (https://www.ipcc-nggip.iges.or.jp/public/2006gl/pdf/0_Overview/V0_1_Overview.pdf#page=6)
- **IPCC Working Group 1 Assessment Report 6**
  - **Summary for Policy Makers** Report (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_SPM.pdf#page=
    - **Figure SPM.4** | Future anthropogenic emissions of key drivers of climate change and warming contributions by groups of drivers for the five illustrative scenarios used in this report. The five scenarios are SSP1-1.9, SSP1-2.6, SSP2-4.5, SSP3-7.0 and SSP5-8.5: Figure SPM.4 (https://www.ipcc.ch/report/ar6/wg1/figures/summary-for-policymakers/figure-spm-4/) Projected CO2 Emissions Dataset (https://dap.ceda.ac.uk/badc/ar6_wg1/data/spm/spm_04/v20210809/panel_a/Carbc Projected NH4 Emissions Dataset (https://dap.ceda.ac.uk/badc/ar6_wg1/data/spm/spm_04/v20210809/panel_a/Metha

Unexplored:

- Various emissions data by country and by source: Integrated Carbon Observation System (https://data.icos-cp.eu/licence?ids=%5B%22G3ZMktDpI9i7qrCXzk4v5C2Z%22%5D&isColl=true)

3.4 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)
- International Institute For Applied System Analysis | Shared Socioeconomic Pathways (SSP) public dataset 2.0: Website (https://tntcat.iiasa.ac.at/SspDb/dsd?Action=htmlpage&page=10)
  - A dataset defining five courses for the world to take regarding economic activity and GHG emissions. Dataset (Requires Registration) (https://tntcat.iiasa.ac.at/SspDb/download/iam_v2/SSP_IAM_V2_201811.csv.zip)
- 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)

Unexplored:

- Another comprehensive emissions dataset: Dataset (https://zenodo.org/record/6483002#.ZC8HAXbMK3B)
- The chart and data motherload: Our World Data (https://ourworldindata.org/energy)
- Energy Consumption Percentage Increases (Will probably go with projections instead): Our World In Data (https://ourworldindata.org/grapher/change-energy-consumption)
- Death Rates by electricity source Our World In Data (https://ourworldindata.org/grapher/death-rates-from-energy-production-per-twh)

3.5 Energy kwh to CO2e Conversion Calculations

- **NREL | Life Cycle Greenhouse Gas Emissions from Electricity Generation: Update (2021)**
  - Table 1. Median Published Life Cycle Emissions Factors for Electricity Generation Technologies, by Life Cycle Phase: NREL (https://www.nrel.gov/docs/fy21osti/80580.pdf#page=3)
    - "We have updated all previous LCAs to use the latest IPCC assessment report, AR5, which increased the GWP for methane to 30 for sources of fossil origin from 25 in AR4 (28).": Report, Page 4 (https://www.pnas.org/doi/epdf/10.1073/pnas.1309334111#page=4)
    - GWP, globalwarming potential harmonized to 100-y IPCC [AR5] values: Report, Page 5 (https://www.pnas.org/doi/epdf/10.1073/pnas.1309334111#page=5)

Unexplored:

- **Climate Change 2014 | Mitigation of Climate Change | Working Group III | Contribution to the Fifth Assessment Report of the IPCC**
  - Table A.III.2 | Emissions of selected electricity supply technologies (gCO2eq/kWh): IPCC report page 7 (https://www.ipcc.ch/site/assets/uploads/2018/02/ipcc_wg3_ar5_annex-iii.pdf#page=7)
- **Renewable Energy Sources and Climate Change Mitigation | Special Report of the IPCC**
- An earlier IPCC report with a number on Oil kwh conversions. IPCC report Figure 9.8, page 732 (https://www.ipcc.ch/site/assets/uploads/2018/03/Chapter-9-Renewable-Energy-in-the-Context-of-Sustainable-Development-1.pdf#page=26) & Table 9.8, page 982 (https://www.ipcc.ch/site/assets/uploads/2018/03/Annex-II-Methodology-1.pdf#page=10)
- **IPCC Fourth Assessment Report | Climate Change 2007 | Working Group I | The Physical Science Basis**
  - Table 2.14. Lifetimes, radiative efficiencies and direct (except for CH4) GWPs relative to CO2: IPCC (https://archive.ipcc.ch/publications_and_data/ar4/wg1/en/ch2s2-10-2.html)

How I found these values:

- Some nuclear site that I found the above link from: Nukes (https://www.world-nuclear.org/information-library/energy-and-the-environment/carbon-dioxide-emissions-from-electricity.aspx)
- US Energy Information's estimate on the same numbers for the USA: EIA (https://www.eia.gov/tools/faqs/faq.php?id=74&t=11)
- BP of all people has some great data

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)
```

6 Data Cleaning

6.1 Loading the Data

6.1.1 Global Temperature

- **IPCC Working Group 1 Assessment Report 6**
  - **Chapter 2 | Changing State of the Climate System**: Report (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_Chapter02.pdf)
    - Figure 2.11 | Annual average global land-ocean surface temperature from 1850–2020 plus or minus the average from 1850–1900 in °C:
      - IPCC Report, Page 316 (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_Chapte Figure (https://www.ipcc.ch/report/ar6/wg1/figures/chapter-2/figure-2-11), References, Page 19 (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_Chapte Dataset (https://dap.ceda.ac.uk/badc/ar6_wg1/data/ch_02/ch2_fig11/v20211207/Figure lower_panel.csv) (Four datasets. The IPCC averages all four together to create Figure 2.11)

In [2]:
```
pd.set_option("display.max_rows", 6)
df_deg_c_raw = pd.read_csv(
    "Data/Figure_2_11c-lower_panel.csv",
    encoding='ANSI')
df_deg_c_raw
```

Out[2]:

	Unnamed: 0	HadCRUT	NOAA	Berkeley	Kadow	Unnamed: 5	HadCRUT confidence limit lower	HadCRUT confidence limit upper
0	1850	-0.063333	-0.019804	-0.069216	-0.082353	NaN	-0.232701	0.110388
1	1851	0.126667	0.060196	0.000784	0.007647	NaN	-0.055366	0.301670
2	1852	0.126667	0.060196	0.040784	0.087647	NaN	-0.052880	0.307086
...	...	...	...	...	...	...	...	...
188	1991-2000	0.673667	0.576196	0.699784	0.683647	NaN	NaN	NaN
189	2001-10	0.912667	0.801196	0.926784	0.904647	NaN	NaN	NaN
190	2011-20	1.115667	1.021196	1.137784	1.087647	NaN	NaN	NaN

191 rows × 8 columns

6.1.2 GHG Atmospheric Concentration

6.1.2.1 Metrcs for GHG

- Table 7.SM.7 | Metrics for greenhouse gases: IPCC (https://github.com/IPCC-WG1/Chapter-7/blob/main/data_output/7sm/metrics_supplement_cleaned.csv)

In [3]:
```
df_metrics_raw = pd.read_csv(
    "Data/metrics_supplement_cleaned.csv",
    encoding='ANSI')
df_metrics_raw
```

Out[3]:

	Name	CAS	Acronym	Formula	Lifetime (yr)	Radiative efficiency (W m-2 ppb-1)	AGWP20 (W m-2 y kg-1)
0	Carbon dioxide	=""	NaN	CO2	NaN	0.000013	2.430000
1	Methane	=""	NaN	CH4	11.800	0.000388	1.980000
2	Nitrous oxide	=""	NaN	N2O	109.000	0.003200	6.650000
...	...	...	...	...	...	...	
246	Octamethylcyclotetrasiloxane	="556-67-2"	NaN	C8H24O4Si4	0.027	0.120000	6.480000
247	Decamethylcyclopentasiloxane	="541-02-6"	NaN	C10H30O5Si5	0.016	0.098000	2.530000
248	Dodecamethylcyclohexasiloxane	="540-97-6"	NaN	C12H36O6Si6	0.011	0.086000	1.240000

249 rows × 18 columns

6.1.2.2 Historical GHG concentrations 1850–2019

- Table AIII.1 | Historical abundances of all GHG: IPCC (https://zenodo.org/record/5705391/files/table_A3.1_historical_abundances.csv)

In [4]:
```
df_hist_ghg_2019_raw = pd.read_csv(
    "Data/table_A3.1_historical_abundances.csv",
    encoding='ANSI', header = [36,37])
df_hist_ghg_2019_raw
```

Out[4]:

	# UNITS:	ppm	ppb		ppt								
	YYYY	CO2	CH4	N2O	HFC-134a	HFC-23	HFC-32	HFC-125	HFC-143a	HFC-152a	HFC-227ea	HFC-236fa	HFC-245fa
0	1750.0	278.3	729.2	270.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	1850.0	285.5	807.6	272.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	1851.0	285.6	807.8	272.2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...
169	2018.0	407.4	1857.8	331.2	101.84	31.16	16.56	26.31	22.42	7.01	1.46	0.18	2.85
170	2019.0	409.9	1866.3	332.1	107.59	32.41	19.98	29.40	24.01	7.13	1.59	0.19	3.06
171	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

172 rows × 61 columns

6.1.2.3 Historical GHG Concentration 1750–2014

- Table T2 | Historical Year 1750–2014: Meinshausen (https://greenhousegases.science.unimelb.edu.au/img/SUPPLEMENT_DataTables_Meinsha

In [5]:
```
df_hist_ghg_2014_raw = pd.read_excel(
    "Data/SUPPLEMENT_DataTables_Meinshausen_6May2020.xlsx",
    header = [8,9,10,11], sheet_name = 'T2 - History Year 1750 to 2014')
df_hist_ghg_2014_raw
```

Out[5]:

	Gas	CO2			CH4			N
	Unit	ppm			ppb			p
	Region	World	Northern Hemisphere	Southern Hemisphere	World	Northern Hemisphere	Southern Hemisphere	W
	Year	Unnamed: 1_level_3	Unnamed: 2_level_3	Unnamed: 3_level_3	Unnamed: 4_level_3	Unnamed: 5_level_3	Unnamed: 6_level_3	U 7
0	1750	277.147003	277.147003	277.147003	731.406006	747.828613	714.983398	2
1	1751	277.187988	277.187988	277.187988	731.838196	747.914124	715.762207	2
2	1752	277.229004	277.229004	277.229004	732.899963	749.263489	716.536438	2
...	...	...	...	...	...	...	...	
262	2012	393.015991	395.036194	390.995789	1815.261230	1861.746582	1768.775879	3
263	2013	395.724976	397.714905	393.735046	1822.580811	1868.968018	1776.193604	3

6.1.2.4 GHG Concentration Projections

- Table AIII.2 | Future abundances of GHG for nine SSP scenarios (2020–2500): Meinshausen (https://greenhousegases.science.unimelb.edu.au/img/SUPPLEMENT_DataTables_Meinsha

In [6]:
```
df_proj_ghg_119_raw = pd.read_excel(
    "Data/SUPPLEMENT_DataTables_Meinshausen_6May2020.xlsx",
    header = [8,9,10,11], sheet_name = 'T3 - SSP1-1.9 ')
df_proj_ghg_119_raw
```

Out[6]:

	Gas	CO2			CH4			N2O
	Unit	ppm			ppb			ppb
	Region	World	Northern Hemisphere	Southern Hemisphere	World	Northern Hemisphere	Southern Hemisphere	Worl
	Year	Unnamed: 1_level_3	Unnamed: 2_level_3	Unnamed: 3_level_3	Unnamed: 4_level_3	Unnamed: 5_level_3	Unnamed: 6_level_3	Unna 7_lev
0	2015	399.949127	401.678070	398.220184	1841.955933	1889.736572	1794.175415	328.
1	2016	403.116974	405.106812	401.127106	1851.593994	1900.575684	1802.612183	329.
2	2017	405.750977	407.749695	403.752228	1872.668823	1919.833740	1825.503906	329.
...	...	...	...	...	...	...	...	
483	2498	336.895996	336.896118	336.895874	871.394897	909.300049	833.489746	357.
484	2499	336.873993	336.874115	336.873871	871.398926	909.304077	833.493774	357.
485	2500	336.863007	336.863129	336.862885	871.400879	909.306091	833.495728	357.

486 rows × 139 columns

In [7]:
```
df_proj_ghg_126_raw = pd.read_excel(
    "Data/SUPPLEMENT_DataTables_Meinshausen_6May2020.xlsx",
    header = [8,9,10,11], sheet_name = 'T4 -  SSP1-2.6 ')
df_proj_ghg_126_raw
```

Out[7]:

	Gas	CO2			CH4			N2O
	Unit	ppm			ppb			ppb
	Region	World	Northern Hemisphere	Southern Hemisphere	World	Northern Hemisphere	Southern Hemisphere	Worl
	Year	Unnamed: 1_level_3	Unnamed: 2_level_3	Unnamed: 3_level_3	Unnamed: 4_level_3	Unnamed: 5_level_3	Unnamed: 6_level_3	Unna 7_lev
0	2015	399.949158	401.678101	398.220215	1841.961060	1889.741577	1794.180542	328.
1	2016	403.116974	405.107117	401.126801	1851.593872	1900.559082	1802.628784	329.
2	2017	405.751953	407.752899	403.751007	1872.298706	1919.339844	1825.257690	329.
...	...	...	...	...	...	...	...	
483	2498	384.325012	384.325134	384.324890	863.948914	901.918457	825.979370	356.
484	2499	384.277985	384.278107	384.277893	863.952881	901.922485	825.983337	356.
485	2500	384.253998	384.254120	384.253876	863.953918	901.923462	825.984375	356.

486 rows × 139 columns

In [8]:
```
df_proj_ghg_245_raw = pd.read_excel(
    "Data/SUPPLEMENT_DataTables_Meinshausen_6May2020.xlsx",
    header = [8,9,10,11], sheet_name = 'T5 - SSP2-4.5 ')
df_proj_ghg_245_raw
```

Out[8]:

	Gas	CO2			CH4			N2O
	Unit	ppm			ppb			ppb
	Region	World	Northern Hemisphere	Southern Hemisphere	World	Northern Hemisphere	Southern Hemisphere	Worl
	Year	Unnamed: 1_level_3	Unnamed: 2_level_3	Unnamed: 3_level_3	Unnamed: 4_level_3	Unnamed: 5_level_3	Unnamed: 6_level_3	Unna 7_lev
0	2015	399.949097	401.678040	398.220154	1841.942627	1889.723267	1794.162109	328.
1	2016	403.116974	405.109192	401.124725	1851.594116	1900.618164	1802.570068	329.
2	2017	405.761963	407.778564	403.745361	1873.799194	1921.280518	1826.317871	329.
...	...	...	...	...	...	...	...	
483	2498	579.432007	579.432129	579.431885	997.319946	1036.824951	957.814880	359.
484	2499	579.263977	579.264099	579.263855	997.310913	1036.816040	957.805908	359.
485	2500	579.179993	579.180115	579.179871	997.305908	1036.811035	957.800903	359.

486 rows × 139 columns

In [9]:
```
df_proj_ghg_370_raw = pd.read_excel(
    "Data/SUPPLEMENT_DataTables_Meinshausen_6May2020.xlsx",
    header = [8,9,10,11], sheet_name = 'T6 - SSP3-7.0 ')
df_proj_ghg_370_raw
```

Out[9]:

	Gas	CO2			CH4			N2
	Unit	ppm			ppb			pp
	Region	World	Northern Hemisphere	Southern Hemisphere	World	Northern Hemisphere	Southern Hemisphere	Wo
	Year	Unnamed: 1_level_3	Unnamed: 2_level_3	Unnamed: 3_level_3	Unnamed: 4_level_3	Unnamed: 5_level_3	Unnamed: 6_level_3	Un 7_l
0	2015	399.948425	401.677399	398.219482	1841.932739	1889.713257	1794.152222	328
1	2016	403.116974	405.118988	401.114960	1851.594116	1900.662231	1802.526123	329
2	2017	405.812958	407.902863	403.723053	1874.509399	1922.320312	1826.698486	329
...	...	...	...	...	...	...	...	
483	2498	1371.869995	1371.870117	1371.869873	1938.680054	1986.280029	1891.080078	515
484	2499	1371.380005	1371.380127	1371.379883	1938.500122	1986.100098	1890.900024	516
485	2500	1371.130005	1371.130127	1371.129883	1938.410034	1986.010010	1890.810181	516

486 rows × 139 columns

In [10]:
```
df_proj_ghg_585_raw = pd.read_excel(
    "Data/SUPPLEMENT_DataTables_Meinshausen_6May2020.xlsx",
    header = [8,9,10,11], sheet_name = 'T11 - SSP5-8.5 ')
df_proj_ghg_585_raw
```

Out[10]:

	Gas	CO2			CH4			N2
	Unit	ppm			ppb			pp
	Region	World	Northern Hemisphere	Southern Hemisphere	World	Northern Hemisphere	Southern Hemisphere	Wo
	Year	Unnamed: 1_level_3	Unnamed: 2_level_3	Unnamed: 3_level_3	Unnamed: 4_level_3	Unnamed: 5_level_3	Unnamed: 6_level_3	Un 7_l
0	2015	399.948792	401.677734	398.219849	1841.947632	1889.728149	1794.167114	328
1	2016	403.116974	405.114441	401.119476	1851.593994	1900.624023	1802.564087	329
2	2017	405.793976	407.849945	403.737976	1873.519165	1921.044922	1825.993408	329
...	...	...	...	...	...	...	...	
483	2498	2011.010010	2011.010132	2011.009888	1019.279968	1060.069458	978.490479	406
484	2499	2010.359985	2010.360107	2010.359863	1019.109924	1059.899414	978.320435	406
485	2500	2010.020020	2010.020142	2010.019897	1019.019958	1059.809326	978.230530	406

486 rows × 139 columns

6.1.2.5 Historical Effective Radiative Forcing All Climate Factors

- Table AIII.3 | Effective radiative forcing (W m^{-2}) time series of all climate forcers from 1750–2019: IPCC (https://zenodo.org/record/5705391/files/table_A3.3_historical_ERF_1750-2019_best_estimate.csv)

In [11]:
```
df_hist_rf_raw = pd.read_csv(
    "Data/table_A3.3_historical_ERF_1750-2019_best_estimate.csv",
    encoding='ANSI')
df_hist_rf_raw
```

Out[11]:

	year	co2	ch4	n2o	other_wmghg	o3	h2o_stratospheric	contrails	a r
0	1750	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.000000	
1	1751	0.001416	0.000508	0.000074	7.958010e-10	0.000300	0.000047	0.000000	
2	1752	0.002832	0.001016	0.000148	1.591602e-09	0.000600	0.000093	0.000000	
...	...	...	...	...	...	...	...	...	
267	2017	2.088686	0.537665	0.202033	4.046273e-01	0.459667	0.049419	0.056725	
268	2018	2.122080	0.540821	0.205654	4.066049e-01	0.467067	0.049709	0.057400	
269	2019	2.156278	0.543983	0.208463	4.080644e-01	0.474467	0.050000	0.057535	

270 rows × 20 columns

6.1.2.6 Historical Minor GHG Breakdon for the Same Time Series Model

- Minor GHG breakdown: IPCC (https://zenodo.org/record/5705391/files/table_A3.3_historical_ERF_1750-2019_minorGHG_breakdown.csv)

In [12]:
```
df_hist_rf_minor_raw = pd.read_csv(
    "Data/table_A3.3_historical_ERF_1750-2019_minorGHG_breakdown.csv",
    encoding='ANSI')
df_hist_rf_minor_raw
```

Out[12]:

	year	HFC-134a	HFC-23	HFC-32	HFC-125	HFC-143a	HFC-152a	HFC-227ea	HFC-236fa	HFC 245f
0	1750	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00(
1	1751	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00(
2	1752	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00(
...	...	...	...	...	...	...	...	...	...	
267	2017	0.016003	0.005729	0.001577	0.005434	0.003500	0.000694	0.000363	0.000042	0.00(
268	2018	0.017022	0.005955	0.001845	0.006151	0.003767	0.000714	0.000398	0.000045	0.00(
269	2019	0.017983	0.006195	0.002227	0.006874	0.004034	0.000726	0.000436	0.000048	0.00(

270 rows × 50 columns

6.1.2.7 Projected Effective Radiative Forcing All Climate Factors

- Table AIII.4 | Effective radiative forcing (W m^{-2}) time series of all climate forcers for nine SSP scenarios (2015–2500): IPCC (https://zenodo.org/record/5705391#.ZDHn1HbMK3A)
  - (Currently, this just links back to the full depository. I'll need to make decisions later about what SSP scenarios to compare my projections to)

In [13]:
```
df_proj_rf_119_raw = pd.read_csv(
    "Data/table_A3.4a_ssp119_ERF_1750-2500_best_estimate.csv",
    encoding='ANSI')
df_proj_rf_119_raw
```

Out[13]:

	year	co2	ch4	n2o	other_wmghg	o3	h2o_stratospheric	contrails	a r
0	1750	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.0	
1	1751	0.001416	0.000508	0.000074	7.958010e-10	0.000300	0.000047	0.0	
2	1752	0.002832	0.001016	0.000148	1.591602e-09	0.000600	0.000093	0.0	
...	...	...	...	...	...	...	...	...	
748	2498	1.053662	0.086960	0.297989	2.417494e-02	0.031792	0.007993	0.0	
749	2499	1.053299	0.086962	0.297967	2.414618e-02	0.031797	0.007993	0.0	
750	2500	1.053117	0.086963	0.297955	2.413171e-02	0.031801	0.007993	0.0	

751 rows × 17 columns

In [14]:
```
df_proj_rf_126_raw = pd.read_csv(
    "Data/table_A3.4b_ssp126_ERF_1750-2500_best_estimate.csv",
    encoding='ANSI')
df_proj_rf_126_raw
```

Out[14]:

	year	co2	ch4	n2o	other_wmghg	o3	h2o_stratospheric	contrails	a r
0	1750	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.0	
1	1751	0.001416	0.000508	0.000074	7.958010e-10	0.000300	0.000047	0.0	
2	1752	0.002832	0.001016	0.000148	1.591602e-09	0.000600	0.000093	0.0	
...	...	...	...	...	...	...	...	...	
748	2498	1.791547	0.082619	0.294730	2.435955e-02	0.011825	0.007594	0.0	
749	2499	1.790857	0.082621	0.294709	2.433219e-02	0.011831	0.007594	0.0	
750	2500	1.790505	0.082622	0.294700	2.431853e-02	0.011837	0.007594	0.0	

751 rows × 17 columns

In [15]:
```
df_proj_rf_245_raw = pd.read_csv(
    "Data/table_A3.4c_ssp245_ERF_1750-2500_best_estimate.csv",
    encoding='ANSI')
df_proj_rf_245_raw
```

Out[15]:

	year	co2	ch4	n2o	other_wmghg	o3	h2o_stratospheric	contrails
0	1750	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.0
1	1751	0.001416	0.000508	0.000074	7.958010e-10	0.000300	0.000047	0.0
2	1752	0.002832	0.001016	0.000148	1.591602e-09	0.000600	0.000093	0.0
...	...	...	...	...	...	...	...	...
748	2498	4.168971	0.157439	0.299972	2.929534e-02	-0.022977	0.014471	0.0
749	2499	4.167244	0.157434	0.299931	2.926934e-02	-0.022976	0.014471	0.0
750	2500	4.166380	0.157432	0.299908	2.925644e-02	-0.022972	0.014470	0.0

751 rows × 17 columns

In [16]:
```
df_proj_rf_370_raw = pd.read_csv(
    "Data/table_A3.4d_ssp370_ERF_1750-2500_best_estimate.csv",
    encoding='ANSI')
df_proj_rf_370_raw
```

Out[16]:

	year	co2	ch4	n2o	other_wmghg	o3	h2o_stratospheric	contrails	a r
0	1750	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.0	
1	1751	0.001416	0.000508	0.000074	7.958010e-10	0.000300	0.000047	0.0	
2	1752	0.002832	0.001016	0.000148	1.591602e-09	0.000600	0.000093	0.0	
...	...	...	...	...	...	...	...	...	
748	2498	9.603838	0.562554	0.699740	5.506137e-02	0.127796	0.051707	0.0	
749	2499	9.601508	0.562488	0.699788	5.502781e-02	0.127734	0.051701	0.0	
750	2500	9.600319	0.562456	0.699813	5.501108e-02	0.127681	0.051698	0.0	

751 rows × 17 columns

In [17]:
```
df_proj_rf_585_raw = pd.read_csv(
    "Data/table_A3.4e_ssp585_ERF_1750-2500_best_estimate.csv",
    encoding='ANSI')
df_proj_rf_585_raw
```

Out[17]:

	year	co2	ch4	n2o	other_wmghg	o3	h2o_stratospheric	contrails
0	1750	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.0
1	1751	0.001416	0.000508	0.000074	7.958010e-10	0.000300	0.000047	0.0
2	1752	0.002832	0.001016	0.000148	1.591602e-09	0.000600	0.000093	0.0
...	...	...	...	...	...	...	...	...
748	2498	12.016389	0.168572	0.414085	5.925787e-02	-0.137316	0.015494	0.0
749	2499	12.014426	0.168482	0.414075	5.922385e-02	-0.137405	0.015486	0.0
750	2500	12.013398	0.168435	0.414070	5.920689e-02	-0.137481	0.015482	0.0

751 rows × 17 columns

6.1.3 Greenhouse Gas Emissions

6.1.3.1 Historical emissions

- Historic emissions data by country by sector: PRIMAP-hist national historical emissions time series (https://zenodo.org/record/7727475#.ZC73QXbMK3A)
  - Key to emissions data: National Greenhouse Gas Inventory, Page 6 (https://www.ipcc-nggip.iges.or.jp/public/2006gl/pdf/0_Overview/V0_1_Overview.pdf#page=6)

In [18]:
```
df_hist_em_raw = pd.read_csv(
"Data/Guetschow-et-al-2023a-PRIMAP-hist_v2.4.2_final_no_rounding_09-Mar-2023.c
    encoding='ANSI')
df_hist_em_raw
```

Out[18]:

	source	scenario (PRIMAP-hist)	area (ISO3)	entity	unit	category (IPCC2006_PRIMAP)	1750	175
0	PRIMAP-hist_v2.4.2_final_nr	HISTCR	ABW	CH4	CH4 * gigagram / a	0	0.012971	0.01
1	PRIMAP-hist_v2.4.2_final_nr	HISTCR	ABW	CH4	CH4 * gigagram / a	1	0.005636	0.00
2	PRIMAP-hist_v2.4.2_final_nr	HISTCR	ABW	CH4	CH4 * gigagram / a	1.A	0.005636	0.00
...	...	...	...	...	...	...	...	
36709	PRIMAP-hist_v2.4.2_final_nr	HISTTP	ZWE	N2O	N2O * gigagram / a	M.AG	0.087681	0.08
36710	PRIMAP-hist_v2.4.2_final_nr	HISTTP	ZWE	N2O	N2O * gigagram / a	M.AG.ELV	0.086519	0.08
36711	PRIMAP-hist_v2.4.2_final_nr	HISTTP	ZWE	N2O	N2O * gigagram / a	M.LULUCF	NaN	

36712 rows × 278 columns

6.1.3.2 Projected Emissions

- **IPCC Working Group 1 Assessment Report 6**
  - **Summary for Policy Makers** Report (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_SPM.pdf#page=
    - **Figure SPM.4** | Future anthropogenic emissions of key drivers of climate change and warming contributions by groups of drivers for the five illustrative scenarios used in this report. The five scenarios are SSP1-1.9, SSP1-2.6, SSP2-4.5, SSP3-7.0 and SSP5-8.5: Figure SPM.4 (https://www.ipcc.ch/report/ar6/wg1/figures/summary-for-policymakers/figure-spm-4/) Projected CO2 Emissions Dataset (https://dap.ceda.ac.uk/badc/ar6_wg1/data/spm/spm_04/v20210809/panel_a/Carbc Projected NH4 Emissions Dataset (https://dap.ceda.ac.uk/badc/ar6_wg1/data/spm/spm_04/v20210809/panel_a/Metha

In [19]:
```
# Though this says it's in Mt of CO2, the report clarifies that it's actually
# in Mt of CH4.

df_proj_em_ch4_raw = pd.read_csv(
    "Data/Methane_Mt_CO2_yr.csv",
    encoding='ANSI')
df_proj_em_ch4_raw
```

Out[19]:

	years	ssp370	ssp119	ssp126	ssp245	ssp585
0	2015	388.072796	388.072796	388.072796	388.072796	388.072796
1	2020	418.454212	358.907755	347.490968	388.090573	397.464394
2	2030	471.330793	243.027020	285.852109	399.446245	454.467560
...	...	...	...	...	...	...
7	2080	693.382453	132.247397	139.312168	302.907284	568.979040
8	2090	735.323193	120.764306	130.129686	298.816324	513.330817
9	2100	777.732192	111.721246	122.195343	295.152937	482.145485

10 rows × 6 columns

In [20]:
```
# Units in Gt_CO2/yr
df_proj_em_co2_raw = pd.read_csv(
    "Data/Carbon_dioxide_Gt_CO2_yr.csv",
    encoding='ANSI')
df_proj_em_co2_raw
```

Out[20]:

	years	ssp370	ssp119	ssp126	ssp245	ssp585
0	2015	39.152726	39.152726	39.152726	39.152726	39.152726
1	2020	44.808038	39.693726	39.804013	40.647530	43.712349
2	2030	52.847359	22.847271	34.734424	43.476063	55.296583
...	...	...	...	...	...	...
7	2080	73.405226	-7.308783	-3.285043	26.838373	129.647035
8	2090	77.799049	-10.565023	-8.385183	16.324392	130.576239
9	2100	82.725833	-13.889788	-8.617786	9.682859	126.287310

10 rows × 6 columns

6.1.4 Energy Production / Consumption

6.1.4.1 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 source: BP (https://www.bp.com/en/global/corporate/energy-economics/statistical-review-of-world-energy.html)

In [21]:
```
df_hist_dem_raw = pd.read_csv(
    "Data/electricity-demand.csv",
    encoding='ANSI')
df_hist_dem_raw
```

Out[21]:

	Entity	Code	Year	Electricity demand (TWh)
0	Afghanistan	AFG	2000	0.57
1	Afghanistan	AFG	2001	0.69
2	Afghanistan	AFG	2002	0.79
...	...	...	...	...
5567	Zimbabwe	ZWE	2019	9.35
5568	Zimbabwe	ZWE	2020	9.58
5569	Zimbabwe	ZWE	2021	9.79

5570 rows × 4 columns

6.1.4.2 Energy Generation

- Energy Production Wattage by Country by Source: Our World In Data (https://ourworldindata.org/grapher/electricity-prod-source-stacked)
  - Original source: BP (https://www.bp.com/en/global/corporate/energy-economics/statistical-review-of-world-energy.html)

In [22]:
```
df_hist_elec_raw = pd.read_csv(
    "Data/electricity-prod-source-stacked.csv",
    encoding='ANSI')
df_hist_elec_raw
```

Out[22]:

	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 from nuclea (TWh)
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	
...	...	...	...	...	...	...	...	...	
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	

10649 rows × 12 columns

6.1.4.3 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 [23]:
```
df_proj_dem_raw = pd.read_csv(
    "Data/IEA-WEO2022_AnnexA_Free_Dataset_Regions.csv",
    encoding='ANSI')
df_proj_dem_raw
```

Out[23]:

	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	
...	...	...	...	...	...	...	...	...	
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

6.1.4.4 Shared Socioeconomic Pathways Energy Projections

- International Institute For Applied System Analysis | Shared Socioeconomic Pathways (SSP) public dataset 2.0: Website (https://tntcat.iiasa.ac.at/SspDb/dsd?Action=htmlpage&page=10)
  - A dataset defining five courses for the world to take regarding economic activity and GHG emissions. Dataset (Requires Registration) (https://tntcat.iiasa.ac.at/SspDb/download/iam_v2/SSP_IAM_V2_201811.csv.zip)

In [24]:
```
df_ssp_proj_raw = pd.read_csv(
    "Data/SSP_IAM_V2_201811.csv",
    encoding='ANSI')
df_ssp_proj_raw
```

Out[24]:

	MODEL	SCENARIO	REGION	VARIABLE	UNIT	2005	2010
0	AIM/CGE	SSP1-19	R5.2ASIA	Agricultural Demand|Crops	million t DM/yr	1045.142000	1127.873
1	AIM/CGE	SSP1-19	R5.2ASIA	Agricultural Demand|Crops|Energy	million t DM/yr	0.000000	5.213
2	AIM/CGE	SSP1-19	R5.2ASIA	Agricultural Demand|Livestock	million t DM/yr	85.998500	94.108
...	...	...	...	...	...	...	
84350	WITCH-GLOBIOM	SSP5-Baseline	World	Secondary Energy|Liquids|Biomass|w/o CCS	EJ/yr	2.288658	4.123
84351	WITCH-GLOBIOM	SSP5-Baseline	World	Secondary Energy|Liquids|Oil	EJ/yr	87.162978	104.367
84352	WITCH-GLOBIOM	SSP5-Baseline	World	Secondary Energy|Solids	EJ/yr	46.720779	46.695

84353 rows × 16 columns

6.1.5 Value counts

In [25]:
```
df_deg_c_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 191 entries, 0 to 190
Data columns (total 8 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Unnamed: 0                      189 non-null    object
 1   HadCRUT                         188 non-null    float64
 2   NOAA                            188 non-null    float64
 3   Berkeley                        188 non-null    float64
 4   Kadow                           188 non-null    float64
 5   Unnamed: 5                      0 non-null      float64
 6   HadCRUT confidence limit lower  171 non-null    float64
 7   HadCRUT confidence limit upper  171 non-null    float64
dtypes: float64(7), object(1)
memory usage: 12.1+ KB
```

In [26]:
```
df_metrics = df_metrics_raw.copy()
df_metrics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 249 entries, 0 to 248
Data columns (total 18 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   Name                                  248 non-null    object 
 1   CAS                                   249 non-null    object 
 2   Acronym                               163 non-null    object 
 3   Formula                               249 non-null    object 
 4   Lifetime (yr)                         248 non-null    float64
 5   Radiative efficiency (W m-2 ppb-1)    249 non-null    float64
 6   AGWP20 (W m-2 yr kg-1)                249 non-null    float64
 7   GWP20                                 249 non-null    float64
 8   AGWP100 (W m-2 yr kg-1)               249 non-null    float64
 9   GWP100                                249 non-null    float64
 10  AGWP500 (W m-2 yr kg-1)               249 non-null    float64
 11  GWP500                                249 non-null    float64
 12  AGTP50 (K kg-1)                       249 non-null    float64
 13  GTP50                                 249 non-null    float64
 14  AGTP100 (K kg-1)                      249 non-null    float64
 15  GTP100                                249 non-null    float64
 16  CGTP50 (yr)                           192 non-null    float64
 17  CGTP100 (yr)                          192 non-null    float64
dtypes: float64(14), object(4)
memory usage: 35.1+ KB
```

In [27]:
```
df_hist_ghg_2019_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 172 entries, 0 to 171
Data columns (total 61 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   (# UNITS:, YYYY)                        171 non-null    float64
 1   (ppm, CO2)                              171 non-null    float64
 2   (ppb, CH4)                              171 non-null    float64
 3   (ppb, N2O)                              171 non-null    float64
 4   (ppt, HFC-134a)                         171 non-null    float64
 5   (ppt, HFC-23)                           171 non-null    float64
 6   (ppt, HFC-32)                           171 non-null    float64
 7   (ppt, HFC-125)                          171 non-null    float64
 8   (ppt, HFC-143a)                         171 non-null    float64
 9   (ppt, HFC-152a)                         171 non-null    float64
 10  (ppt, HFC-227ea)                        171 non-null    float64
 11  (ppt, HFC-236fa)                        171 non-null    float64
 12  (ppt, HFC-245fa)                        171 non-null    float64
 13  (ppt, HFC-365mfc)                       171 non-null    float64
 14  (ppt, HFC-43-10mee)                     171 non-null    float64
 15  (ppt, NF3)                              171 non-null    float64
 16  (ppt, SF6)                              171 non-null    float64
 17  (ppt, SO2F2)                            171 non-null    float64
 18  (ppt, CF4)                              171 non-null    float64
 19  (ppt, C2F6)                             171 non-null    float64
 20  (ppt, C3F8)                             171 non-null    float64
 21  (ppt, c-C4F8)                           171 non-null    float64
 22  (ppt, CFC-12)                           171 non-null    float64
 23  (ppt, CFC-11)                           171 non-null    float64
 24  (ppt, CFC-113)                          171 non-null    float64
 25  (ppt, CFC-114)                          171 non-null    float64
 26  (ppt, CFC-115)                          171 non-null    float64
 27  (ppt, CFC-13)                           171 non-null    float64
 28  (ppt, HCFC-22)                          171 non-null    float64
 29  (ppt, HCFC-141b)                        171 non-null    float64
 30  (ppt, HCFC-142b)                        171 non-null    float64
 31  (ppt, CH3CCl3)                          171 non-null    float64
 32  (ppt, CCl4)                             171 non-null    float64
 33  (ppt, CH3Cl)                            171 non-null    float64
 34  (ppt, CH3Br)                            171 non-null    float64
 35  (ppt, CH2Cl2)                           171 non-null    float64
 36  (ppt, CHCl3)                            171 non-null    float64
 37  (ppt, Halon-1211)                       171 non-null    float64
 38  (ppt, Halon-1301)                       171 non-null    float64
 39  (ppt, Halon-2402)                       171 non-null    float64
 40  (Unnamed: 40_level_0, n-C4F10)          167 non-null    float64
 41  (Unnamed: 41_level_0, n-C5F12)          167 non-null    float64
 42  (Unnamed: 42_level_0, n-C6F14)          167 non-null    float64
 43  (Unnamed: 43_level_0, i-C6F14)          5 non-null      float64
 44  (Unnamed: 44_level_0, C7F16)            167 non-null    float64
 45  (Unnamed: 45_level_0, C8F18)            166 non-null    float64
 46  (Unnamed: 46_level_0, CFC-112)          39 non-null     float64
 47  (Unnamed: 47_level_0, CFC-112a)         40 non-null     float64
 48  (Unnamed: 48_level_0, CFC-113a)         39 non-null     float64
 49  (Unnamed: 49_level_0, CFC-114a)         40 non-null     float64
 50  (Unnamed: 50_level_0, HCFC-133a)        41 non-null     float64
 51  (Unnamed: 51_level_0, HCFC-31)          16 non-null     float64
```

```
 52  (Unnamed: 52_level_0, HCFC-124)             13 non-null     float64
 53  (Unnamed: 53_level_0, Unnamed: 53_level_1)  0 non-null      float64
 54  (Unnamed: 54_level_0, Unnamed: 54_level_1)  0 non-null      float64
 55  (Unnamed: 55_level_0, Unnamed: 55_level_1)  0 non-null      float64
 56  (Unnamed: 56_level_0, Unnamed: 56_level_1)  0 non-null      float64
 57  (Unnamed: 57_level_0, Unnamed: 57_level_1)  0 non-null      float64
 58  (Unnamed: 58_level_0, Unnamed: 58_level_1)  0 non-null      float64
 59  (Unnamed: 59_level_0, Unnamed: 59_level_1)  0 non-null      float64
 60  (Unnamed: 60_level_0, Unnamed: 60_level_1)  0 non-null      float64
dtypes: float64(61)
memory usage: 82.1 KB
```

In [28]:
```
df_hist_ghg_2014_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 265 entries, 0 to 264
Columns: 139 entries, ('Gas', 'Unit', 'Region', 'Year') to ('Halon 2402', 'pp
t', 'Southern Hemisphere', 'Unnamed: 138_level_3')
dtypes: float64(138), int64(1)
memory usage: 287.9 KB
```

In [29]:
```
df_proj_ghg_119_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 486 entries, 0 to 485
Columns: 139 entries, ('Gas', 'Unit', 'Region', 'Year') to ('Halon 2402', 'pp
t', 'Southern Hemisphere', 'Unnamed: 138_level_3')
dtypes: float64(138), int64(1)
memory usage: 527.9 KB
```

In [30]:
```
df_hist_rf_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 20 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   year                              270 non-null    int64  
 1   co2                               270 non-null    float64
 2   ch4                               270 non-null    float64
 3   n2o                               270 non-null    float64
 4   other_wmghg                       270 non-null    float64
 5   o3                                270 non-null    float64
 6   h2o_stratospheric                 270 non-null    float64
 7   contrails                         270 non-null    float64
 8   aerosol-radiation_interactions    270 non-null    float64
 9   aerosol-cloud_interactions        270 non-null    float64
 10  bc_on_snow                        270 non-null    float64
 11  land_use                          270 non-null    float64
 12  volcanic                          270 non-null    float64
 13  solar                             270 non-null    float64
 14  nonco2_wmghg                      270 non-null    float64
 15  aerosol                           270 non-null    float64
 16  chapter2_other_anthro             270 non-null    float64
 17  total_anthropogenic               270 non-null    float64
 18  total_natural                     270 non-null    float64
 19  total                             270 non-null    float64
dtypes: float64(19), int64(1)
memory usage: 42.3 KB
```

In [31]:
```
df_hist_rf_minor_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 50 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   year          270 non-null    int64  
 1   HFC-134a      270 non-null    float64
 2   HFC-23        270 non-null    float64
 3   HFC-32        270 non-null    float64
 4   HFC-125       270 non-null    float64
 5   HFC-143a      270 non-null    float64
 6   HFC-152a      270 non-null    float64
 7   HFC-227ea     270 non-null    float64
 8   HFC-236fa     270 non-null    float64
 9   HFC-245fa     270 non-null    float64
 10  HFC-365mfc    270 non-null    float64
 11  HFC-43-10mee  270 non-null    float64
 12  NF3           270 non-null    float64
 13  SF6           270 non-null    float64
 14  SO2F2         270 non-null    float64
 15  CF4           270 non-null    float64
 16  C2F6          270 non-null    float64
 17  C3F8          270 non-null    float64
 18  c-C4F8        270 non-null    float64
 19  CFC-12        270 non-null    float64
 20  CFC-11        270 non-null    float64
 21  CFC-113       270 non-null    float64
 22  CFC-114       270 non-null    float64
 23  CFC-115       270 non-null    float64
 24  CFC-13        270 non-null    float64
 25  HCFC-22       270 non-null    float64
 26  HCFC-141b     270 non-null    float64
 27  HCFC-142b     270 non-null    float64
 28  CH3CCl3       270 non-null    float64
 29  CCl4          270 non-null    float64
 30  CH3Cl         270 non-null    float64
 31  CH3Br         270 non-null    float64
 32  CH2Cl2        270 non-null    float64
 33  CHCl3         270 non-null    float64
 34  Halon-1211    270 non-null    float64
 35  Halon-1301    270 non-null    float64
 36  Halon-2402    270 non-null    float64
 37  n-C4F10       270 non-null    float64
 38  n-C5F12       270 non-null    float64
 39  n-C6F14       270 non-null    float64
 40  i-C6F14       270 non-null    float64
 41  C7F16         270 non-null    float64
 42  C8F18         270 non-null    float64
 43  CFC-112       270 non-null    float64
 44  CFC-112a      270 non-null    float64
 45  CFC-113a      270 non-null    float64
 46  CFC-114a      270 non-null    float64
 47  HCFC-133a     270 non-null    float64
 48  HCFC-31       270 non-null    float64
 49  HCFC-124      270 non-null    float64
```

```
dtypes: float64(49), int64(1)
memory usage: 105.6 KB
```

In [32]:
```
df_hist_em_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36712 entries, 0 to 36711
Columns: 278 entries, source to 2021
dtypes: float64(272), object(6)
memory usage: 77.9+ MB
```

In [33]:
```
df_hist_em_raw.columns
```

Out[33]:
```
Index(['source', 'scenario (PRIMAP-hist)', 'area (ISO3)', 'entity', 'unit',
       'category (IPCC2006_PRIMAP)', '1750', '1751', '1752', '1753',
       ...
       '2012', '2013', '2014', '2015', '2016', '2017', '2018', '2019', '202
0',
       '2021'],
      dtype='object', length=278)
```

In [34]:
```
df_hist_em_raw[['source', 'scenario (PRIMAP-hist)', 'area (ISO3)', 'entity',
                'unit', 'category (IPCC2006_PRIMAP)', '1750', '2021']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36712 entries, 0 to 36711
Data columns (total 8 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   source                      36712 non-null  object
 1   scenario (PRIMAP-hist)      36712 non-null  object
 2   area (ISO3)                 36712 non-null  object
 3   entity                      36712 non-null  object
 4   unit                        36712 non-null  object
 5   category (IPCC2006_PRIMAP)  36712 non-null  object
 6   1750                        36013 non-null  float64
 7   2021                        36712 non-null  float64
dtypes: float64(2), object(6)
memory usage: 2.2+ MB
```

In [35]:
```
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 [36]:
```
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 [37]:
```
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
```

In [38]:
```
df_ssp_proj_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84353 entries, 0 to 84352
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   MODEL     84353 non-null  object 
 1   SCENARIO  84353 non-null  object 
 2   REGION    84353 non-null  object 
 3   VARIABLE  84353 non-null  object 
 4   UNIT      84353 non-null  object 
 5   2005      67962 non-null  float64
 6   2010      83666 non-null  float64
 7   2020      84227 non-null  float64
 8   2030      84227 non-null  float64
 9   2040      84227 non-null  float64
 10  2050      84224 non-null  float64
 11  2060      84224 non-null  float64
 12  2070      84224 non-null  float64
 13  2080      84215 non-null  float64
 14  2090      84215 non-null  float64
 15  2100      84215 non-null  float64
dtypes: float64(11), object(5)
memory usage: 10.3+ MB
```

6.1.6 Rename Columns

The columns need more succinct names.

6.1.6.1 Renaming df_deg_c

In [39]:
```
df_deg_c = df_deg_c_raw.copy()

pd.set_option("display.max_rows", 100)

df_deg_c = df_deg_c_raw.copy()

renaming = {'Unnamed: 0' : 'Year', 'Unnamed: 5' :
            'Global Annual Average Anomaly (°C)'}
df_deg_c.rename(columns = renaming, inplace = True)

# The two columns on the right need to be dropped, and rows 171-190 need to be
# dropped as they are a new table, not a continuation.

df_deg_c.drop(columns = ['HadCRUT confidence limit lower',
                         'HadCRUT confidence limit upper'],
              index = range(171,191), inplace=True)

df_deg_c['Global Annual Average Anomaly (°C)'] = df_deg_c.iloc[:,1:].mean(axis

df_deg_c['Year'] = df_deg_c['Year'].astype(int)
df_deg_c.index = df_deg_c['Year']
df_deg_c.drop(columns = 'Year', inplace = True)
df_deg_c
```

Out[39]:

Year	HadCRUT	NOAA	Berkeley	Kadow	Global Annual Average Anomaly (°C)
1850	-0.063333	-0.019804	-0.069216	-0.082353	-0.058676
1851	0.126667	0.060196	0.000784	0.007647	0.048824
1852	0.126667	0.060196	0.040784	0.087647	0.078824
1853	0.086667	0.040196	0.030784	0.027647	0.046324
1854	0.066667	0.050196	0.020784	0.057647	0.048824
...	...	...	...	...	...
2016	1.286667	1.200196	1.320784	1.247647	1.263824
2017	1.206667	1.120196	1.230784	1.157647	1.178824
2018	1.116667	1.040196	1.150784	1.097647	1.101324
2019	1.246667	1.150196	1.290784	1.217647	1.226324
2020	1.276667	1.190196	1.310784	1.247647	1.256324

171 rows × 5 columns

6.1.6.2 Renaming df_hist_ghg_2019

In [40]:
```
df_hist_ghg_2019 = df_hist_ghg_2019_raw.copy()

# The original data has some empty rows, so I'll delete those.

df_hist_ghg_2019 = df_hist_ghg_2019.iloc[0:171,0:53]

df_hist_ghg_2019.columns = [col[1] + ' (' + col[0] + ')' for col in
                        df_hist_ghg_2019.columns.values]
renaming = {'YYYY (# UNITS:)': 'Year',
            'n-C4F10 (Unnamed: 40_level_0)'   : 'n-C4F10 (ppt)',
            'n-C5F12 (Unnamed: 41_level_0)'   : 'n-C5F12 (ppt)',
            'n-C6F14 (Unnamed: 42_level_0)'   : 'n-C6F14 (ppt)',
            'i-C6F14 (Unnamed: 43_level_0)'   : 'i-C6F14 (ppt)',
            'C7F16 (Unnamed: 44_level_0)'     : 'C7F16 (ppt)',
            'C8F18 (Unnamed: 45_level_0)'     : 'C8F18 (ppt)',
            'CFC-112 (Unnamed: 46_level_0)'   : 'CFC-112 (ppt)',
            'CFC-112a (Unnamed: 47_level_0)'  : 'CFC-112a (ppt)',
            'CFC-113a (Unnamed: 48_level_0)'  : 'CFC-113a (ppt)',
            'CFC-114a (Unnamed: 49_level_0)'  : 'CFC-114a (ppt)',
            'HCFC-133a (Unnamed: 50_level_0)' : 'HCFC-133a (ppt)',
            'HCFC-31 (Unnamed: 51_level_0)'   : 'HCFC-31 (ppt)',
            'HCFC-124 (Unnamed: 52_level_0)'  : 'HCFC-124 (ppt)}'}

df_hist_ghg_2019.rename(columns = renaming, inplace = True)
df_hist_ghg_2019['Year'] = df_hist_ghg_2019['Year'].astype(int)
df_hist_ghg_2019.index = df_hist_ghg_2019['Year']
df_hist_ghg_2019 = df_hist_ghg_2019[['CO2 (ppm)', 'CH4 (ppb)']]

df_hist_ghg_2019
```

Out[40]:

Year	CO2 (ppm)	CH4 (ppb)
1750	278.3	729.2
1850	285.5	807.6
1851	285.6	807.8
1852	285.8	808.5
1853	285.9	809.8
...	...	...
2015	399.4	1834.0
2016	402.9	1842.2
2017	405.0	1849.2
2018	407.4	1857.8

6.1.6.3 Renaming df_hist_ghg_2014

In [41]:
```
df_hist_ghg_2014 = df_hist_ghg_2014_raw.copy()

# Three columns are needed from this DataFrame, Year, CO2 and NH4 emissions.
# Also, there's no need for data before 2020 because I already have historical
# data for that set.
df_hist_ghg_2014 = df_hist_ghg_2014.iloc[:,[0,1,4]]

# Getting rid of multilevel index names for columns
df_hist_ghg_2014.columns = [col[2] + ' ' + col[0] + ' (' + col[1] + ')'
                                    for col in df_hist_ghg_2014.columns.values]

# Renaming the first column because it is wrong.
renaming = {'Region Gas (Unit)' : 'Year',
            'World CO2 (ppm)' : 'CO2 (ppm)',
            'World CH4 (ppb)' : 'CH4 (ppb)'}
df_hist_ghg_2014.rename(columns = renaming, inplace = True)

df_hist_ghg_2014['Year'] = df_hist_ghg_2014['Year'].astype(int)
df_hist_ghg_2014.index = df_hist_ghg_2014['Year']
df_hist_ghg_2014.drop(columns = 'Year', inplace = True)

df_hist_ghg_2014
```

Out[41]:

Year	CO2 (ppm)	CH4 (ppb)
1750	277.147003	731.406006
1751	277.187988	731.838196
1752	277.229004	732.899963
1753	277.263000	733.634033
1754	277.303986	734.202026
...	...	...
2010	388.717041	1807.850708
2011	390.944000	1813.070190
2012	393.015991	1815.261230
2013	395.724976	1822.580811
2014	397.546967	1831.470947

265 rows × 2 columns

6.1.6.4 Renaming df_proj_ghg_119

In [42]:
```
df_proj_ghg_119 = df_proj_ghg_119_raw.copy()

# Three columns are needed from this DataFrame, Year, CO2 and NH4 emissions.
# Also, there's no need for data before 2020 because I already have historical
# data for that set.
df_proj_ghg_119 = df_proj_ghg_119.iloc[:,[0,1,4]]

# Getting rid of multilevel index names for columns
df_proj_ghg_119.columns = ['SSP1-1.9 ' + col[0] + ' (' + col[1] + ')'
                                    for col in df_proj_ghg_119.columns.values]

# Renaming the first column because it is wrong.
renaming = {'SSP1-1.9 Gas (Unit)' : 'Year'}

df_proj_ghg_119.rename(columns = renaming, inplace = True)

df_proj_ghg_119['Year'] = df_proj_ghg_119['Year'].astype(int)
df_proj_ghg_119.index = df_proj_ghg_119['Year']
df_proj_ghg_119.drop(columns = 'Year', inplace = True)

df_proj_ghg_119
```

Out[42]:

Year	SSP1-1.9 CO2 (ppm)	SSP1-1.9 CH4 (ppb)
2015	399.949127	1841.955933
2016	403.116974	1851.593994
2017	405.750977	1872.668823
2018	408.586975	1882.600586
2019	411.404968	1889.650024
...	...	...
2496	336.940002	871.387878
2497	336.917999	871.390930
2498	336.895996	871.394897
2499	336.873993	871.398926

6.1.6.5 Renaming df_proj_ghg_126

In [43]:
```
df_proj_ghg_126 = df_proj_ghg_126_raw.copy()

# Three columns are needed from this DataFrame, Year, CO2 and NH4 emissions.
# Also, there's no need for data before 2020 because I already have historical
# data for that set.
df_proj_ghg_126 = df_proj_ghg_126.iloc[:,[0,1,4]]

# Getting rid of multilevel index names for columns
df_proj_ghg_126.columns = ['SSP1-2.6 ' + col[0] + ' (' + col[1] + ')'
                                    for col in df_proj_ghg_126.columns.values]

# Renaming the first column because it is wrong.
renaming = {'SSP1-2.6 Gas (Unit)' : 'Year'}

df_proj_ghg_126.rename(columns = renaming, inplace = True)

df_proj_ghg_126['Year'] = df_proj_ghg_126['Year'].astype(int)
df_proj_ghg_126.index = df_proj_ghg_126['Year']
df_proj_ghg_126.drop(columns = 'Year', inplace = True)

df_proj_ghg_126
```

Out[43]:

Year	SSP1-2.6 CO2 (ppm)	SSP1-2.6 CH4 (ppb)
2015	399.949158	1841.961060
2016	403.116974	1851.593872
2017	405.751953	1872.298706
2018	408.591949	1881.130371
2019	411.416962	1886.409912
...	...	...
2496	384.419006	863.942932
2497	384.372009	863.945923
2498	384.325012	863.948914
2499	384.277985	863.952881

6.1.6.6 Renaming df_proj_ghg_245

In [44]:
```
df_proj_ghg_245 = df_proj_ghg_245_raw.copy()

# Three columns are needed from this DataFrame, Year, CO2 and NH4 emissions.
# Also, there's no need for data before 2020 because I already have historical
# data for that set.
df_proj_ghg_245 = df_proj_ghg_245.iloc[:,[0,1,4]]

# Getting rid of multilevel index names for columns
df_proj_ghg_245.columns = ['SSP2-4.5 ' + col[0] + ' (' + col[1] + ')'
                                    for col in df_proj_ghg_245.columns.values]

# Renaming the first column because it is wrong.
renaming = {'SSP2-4.5 Gas (Unit)' : 'Year'}

df_proj_ghg_245.rename(columns = renaming, inplace = True)

df_proj_ghg_245['Year'] = df_proj_ghg_245['Year'].astype(int)
df_proj_ghg_245.index = df_proj_ghg_245['Year']
df_proj_ghg_245.drop(columns = 'Year', inplace = True)

df_proj_ghg_245
```

Out[44]:

Year	SSP2-4.5 CO2 (ppm)	SSP2-4.5 CH4 (ppb)
2015	399.949097	1841.942627
2016	403.116974	1851.594116
2017	405.761963	1873.799194
2018	408.631958	1887.041016
2019	411.505951	1899.410645
...	...	...
2496	579.768982	997.339905
2497	579.599976	997.329956
2498	579.432007	997.319946
2499	579.263977	997.310913

6.1.6.7 Renaming df_proj_ghg_370

In [45]:
```
df_proj_ghg_370 = df_proj_ghg_370_raw.copy()

# Three columns are needed from this DataFrame, Year, CO2 and NH4 emissions.
# Also, there's no need for data before 2020 because I already have historical
# data for that set.
df_proj_ghg_370 = df_proj_ghg_370.iloc[:,[0,1,4]]

# Getting rid of multilevel index names for columns
df_proj_ghg_370.columns = ['SSP3-7.0 ' + col[0] + ' (' + col[1] + ')'
                                    for col in df_proj_ghg_370.columns.values]

# Renaming the first column because it is wrong.
renaming = {'SSP3-7.0 Gas (Unit)' : 'Year'}

df_proj_ghg_370.rename(columns = renaming, inplace = True)

df_proj_ghg_370['Year'] = df_proj_ghg_370['Year'].astype(int)
df_proj_ghg_370.index = df_proj_ghg_370['Year']
df_proj_ghg_370.drop(columns = 'Year', inplace = True)

df_proj_ghg_370
```

Out[45]:

Year	SSP3-7.0 CO2 (ppm)	SSP3-7.0 CH4 (ppb)
2015	399.948425	1841.932739
2016	403.116974	1851.594116
2017	405.812958	1874.509399
2018	408.831970	1889.791260
2019	411.945953	1905.450928
...	...	...
2496	1372.869995	1939.030029
2497	1372.369995	1938.860107
2498	1371.869995	1938.680054
2499	1371.380005	1938.500122

6.1.6.8 Renaming df_proj_ghg_585

In [46]:
```
df_proj_ghg_585 = df_proj_ghg_585_raw.copy()

# Three columns are needed from this DataFrame, Year, CO2 and NH4 emissions.
# Also, there's no need for data before 2020 because I already have historical
# data for that set.
df_proj_ghg_585 = df_proj_ghg_585.iloc[:,[0,1,4]]

# Getting rid of multilevel index names for columns
df_proj_ghg_585.columns = ['SSP5-8.5 ' + col[0] + ' (' + col[1] + ')'
                                    for col in df_proj_ghg_585.columns.values]

# Renaming the first column because it is wrong.
renaming = {'SSP5-8.5 Gas (Unit)' : 'Year'}

df_proj_ghg_585.rename(columns = renaming, inplace = True)

df_proj_ghg_585['Year'] = df_proj_ghg_585['Year'].astype(int)
df_proj_ghg_585.index = df_proj_ghg_585['Year']
df_proj_ghg_585.drop(columns = 'Year', inplace = True)

df_proj_ghg_585
```

Out[46]:

Year	SSP5-8.5 CO2 (ppm)	SSP5-8.5 CH4 (ppb)
2015	399.948792	1841.947632
2016	403.116974	1851.593994
2017	405.793976	1873.519165
2018	408.759949	1885.970947
2019	411.789948	1897.060425
...	...	...
2496	2012.339966	1019.619934
2497	2011.670044	1019.449951
2498	2011.010010	1019.279968
2499	2010.359985	1019.109924

6.1.6.9 Renaming df_hist_rf

In [47]:
```
def rf_clean(raw, name):
    df = raw.copy()

    renaming = {'year' : 'Year',
            'co2' : name + ' CO2 (W/m²)',
            'ch4' : name + ' CH4 (W/m²)',
            'n2o' : name + ' N2O (W/m²)',
            'other_wmghg' : name + ' Minor GHG (W/m²)',
            'o3' : name + ' O3 (W/m²)',
            'h2o_stratospheric' : name + ' Stratospheric H2O (W/m²)',
            'contrails' : name + ' Contrails (W/m²)',
            'aerosol-radiation_interactions' : name + ' Aerosol Radiation (W/m
            'aerosol-cloud_interactions' : name + ' Aerosol Cloud (W/m²)',
            'bc_on_snow' : name + ' Snow (W/m²)',
            'land_use' : name + ' Land Use (W/m²)',
            'volcanic' : name + ' Volcanic (W/m²)',
            'solar' : name + ' Solar (W/m²)',
            'nonco2_wmghg' : name + ' NonCO2 GHG (W/m²)',
            'aerosol' : name + ' Aerosol (W/m²)',
            'chapter2_other_anthro' : name + ' Ch2 Other Anthro (W/m²)',
            'total_anthropogenic' : name + ' Total Antropgenic (W/m²)',
            'total_natural' : name + ' Total Natural (W/m²)',
            'total' : name + ' Total (W/m²)'}

    df.rename(columns = renaming, inplace = True)
    df['Year'] = df['Year'].astype(int)
    df.index = df['Year']
    df.drop(columns = 'Year', inplace = True)
    return df

df_hist_rf = rf_clean(df_hist_rf_raw, 'Hist')
df_hist_rf = df_hist_rf.iloc[:,[18, 0, 1, 2, 3, 4, 5, 6,
                                7, 8, 9, 10, 11, 12, 16, 17]]

df_hist_rf
```

Out[47]:

Year	Hist Total (W/m²)	Hist CO2 (W/m²)	Hist CH4 (W/m²)	Hist N2O (W/m²)	Hist Minor GHG (W/m²)	Hist O3 (W/m²)	Hist Stratospheric H2O (W/m²)	Hist Contrail (W/m²)
1750	0.297568	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.00000
1751	0.286642	0.001416	0.000508	0.000074	7.958010e-10	0.000300	0.000047	0.00000
1752	0.261955	0.002832	0.001016	0.000148	1.591602e-09	0.000600	0.000093	0.00000
1753	0.224661	0.004247	0.001524	0.000222	2.387403e-09	0.000900	0.000140	0.00000
1754	0.184225	0.005662	0.002031	0.000296	3.183204e-09	0.001200	0.000187	0.00000
...	...	...	...	...	...	...	...	.
2015	2.612136	2.010013	0.531999	0.196590	4.008651e-01	0.467400	0.048899	0.05099
2016	2.676478	2.058493	0.535044	0.199149	4.026474e-01	0.463533	0.049178	0.05342

6.1.6.10 Renaming df_proj_rf_119

In [48]:
```
df_proj_rf_119 = rf_clean(df_proj_rf_119_raw, 'SSP1-1.9')
df_proj_rf_119 = df_proj_rf_119.iloc[:,[15, 0, 1, 2, 3, 4, 5, 6,
                                        7, 8, 9, 10, 11, 12, 13, 14]]
df_proj_rf_119
```

Out[48]:

	SSP1-1.9 Total (W/m²)	SSP1-1.9 CO2 (W/m²)	SSP1-1.9 CH4 (W/m²)	SSP1-1.9 N2O (W/m²)	SSP1-1.9 Minor GHG (W/m²)	SSP1-1.9 O3 (W/m²)	SSP1-1.9 Stratospheric H2O (W/m²)	SSP1-1. Contrail (W/m²)
Year								
1750	0.297568	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.
1751	0.286266	0.001416	0.000508	0.000074	7.958010e-10	0.000300	0.000047	0.
1752	0.261437	0.002832	0.001016	0.000148	1.591602e-09	0.000600	0.000093	0.
1753	0.224236	0.004247	0.001524	0.000222	2.387403e-09	0.000900	0.000140	0.
1754	0.184178	0.005662	0.002031	0.000296	3.183204e-09	0.001200	0.000187	0.
...	...	...	...	...	...	...	...	.
2496	1.378572	1.054389	0.086955	0.298034	2.423314e-02	0.031784	0.007992	0.
2497	1.378164	1.054026	0.086957	0.298012	2.420396e-02	0.031788	0.007993	0.

6.1.6.11 Renaming df_proj_rf_126

In [49]:
```
df_proj_rf_126 = rf_clean(df_proj_rf_126_raw, 'SSP1-2.6')
df_proj_rf_126 = df_proj_rf_126.iloc[:,[15, 0, 1, 2, 3, 4, 5, 6,
                                        7, 8, 9, 10, 11, 12, 13, 14]]
df_proj_rf_126
```

Out[49]:

	SSP1-2.6 Total (W/m²)	SSP1-2.6 CO2 (W/m²)	SSP1-2.6 CH4 (W/m²)	SSP1-2.6 N2O (W/m²)	SSP1-2.6 Minor GHG (W/m²)	SSP1-2.6 O3 (W/m²)	SSP1-2.6 Stratospheric H2O (W/m²)	SSP1-2. Contrail (W/m²)
Year								
1750	0.297568	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.
1751	0.286266	0.001416	0.000508	0.000074	7.958010e-10	0.000300	0.000047	0.
1752	0.261437	0.002832	0.001016	0.000148	1.591602e-09	0.000600	0.000093	0.
1753	0.224236	0.004247	0.001524	0.000222	2.387403e-09	0.000900	0.000140	0.
1754	0.184178	0.005662	0.002031	0.000296	3.183204e-09	0.001200	0.000187	0.
...	...	...	...	...	...	...	...	.
2496	2.107866	1.792927	0.082615	0.294776	2.441499e-02	0.011815	0.007594	0.
2497	2.107131	1.792237	0.082617	0.294752	2.438718e-02	0.011820	0.007594	0.

6.1.6.12 Renaming df_proj_rf_245

In [50]:
```
df_proj_rf_245 = rf_clean(df_proj_rf_245_raw, 'SSP2-4.5')
df_proj_rf_245 = df_proj_rf_245.iloc[:,[15, 0, 1, 2, 3, 4, 5, 6,
                                        7, 8, 9, 10, 11, 12, 13, 14]]
df_proj_rf_245
```

Out[50]:

	SSP2-4.5 Total (W/m²)	SSP2-4.5 CO2 (W/m²)	SSP2-4.5 CH4 (W/m²)	SSP2-4.5 N2O (W/m²)	SSP2-4.5 Minor GHG (W/m²)	SSP2-4.5 O3 (W/m²)	SSP2-4.5 Stratospheric H2O (W/m²)	SSP2-4 Contrai (W/m²)
Year								
1750	0.297568	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0
1751	0.286266	0.001416	0.000508	0.000074	7.958010e-10	0.000300	0.000047	0
1752	0.261437	0.002832	0.001016	0.000148	1.591602e-09	0.000600	0.000093	0
1753	0.224236	0.004247	0.001524	0.000222	2.387403e-09	0.000900	0.000140	0
1754	0.184178	0.005662	0.002031	0.000296	3.183204e-09	0.001200	0.000187	0
...	...	...	...	...	...	...	...	
2496	4.476728	4.172433	0.157449	0.300056	2.934801e-02	-0.022979	0.014472	0
2497	4.474916	4.170697	0.157444	0.300012	2.932158e-02	-0.022978	0.014471	0

6.1.6.13 Renaming df_proj_rf_370

In [51]:
```
df_proj_rf_370 = rf_clean(df_proj_rf_370_raw, 'SSP3-7.0')
df_proj_rf_370 = df_proj_rf_370.iloc[:,[15, 0, 1, 2, 3, 4, 5, 6,
                                        7, 8, 9, 10, 11, 12, 13, 14]]
df_proj_rf_370
```

Out[51]:

	SSP3-7.0 Total (W/m²)	SSP3-7.0 CO2 (W/m²)	SSP3-7.0 CH4 (W/m²)	SSP3-7.0 N2O (W/m²)	SSP3-7.0 Minor GHG (W/m²)	SSP3-7.0 O3 (W/m²)	SSP3-7.0 Stratospheric H2O (W/m²)	SSP3-7 Contra (W/m²)
Year								
1750	0.297568	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	(
1751	0.286266	0.001416	0.000508	0.000074	7.958010e-10	0.000300	0.000047	(
1752	0.261437	0.002832	0.001016	0.000148	1.591602e-09	0.000600	0.000093	(
1753	0.224236	0.004247	0.001524	0.000222	2.387403e-09	0.000900	0.000140	(
1754	0.184178	0.005662	0.002031	0.000296	3.183204e-09	0.001200	0.000187	(
...	...	...	...	...	...	...	...	
2496	10.691539	9.608590	0.562681	0.699643	5.512898e-02	0.127919	0.051719	(
2497	10.689050	9.606215	0.562619	0.699691	5.509505e-02	0.127859	0.051713	(

6.1.6.14 Renaming df_proj_rf_585

In [52]:
```
df_proj_rf_585 = rf_clean(df_proj_rf_585_raw, 'SSP5-8.5')
df_proj_rf_585 = df_proj_rf_585.iloc[:,[15, 0, 1, 2, 3, 4, 5, 6,
                                        7, 8, 9, 10, 11, 12, 13, 14]]
df_proj_rf_585
```

Out[52]:

Year	SSP5-8.5 Total (W/m²)	SSP5-8.5 CO2 (W/m²)	SSP5-8.5 CH4 (W/m²)	SSP5-8.5 N2O (W/m²)	SSP5-8.5 Minor GHG (W/m²)	SSP5-8.5 O3 (W/m²)	SSP5-8.5 Stratospheric H2O (W/m²)	SSP5 Cont (W/m
1750	0.297568	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	
1751	0.286266	0.001416	0.000508	0.000074	7.958010e-10	0.000300	0.000047	
1752	0.261437	0.002832	0.001016	0.000148	1.591602e-09	0.000600	0.000093	
1753	0.224236	0.004247	0.001524	0.000222	2.387403e-09	0.000900	0.000140	
1754	0.184178	0.005662	0.002031	0.000296	3.183204e-09	0.001200	0.000187	
...	...	...	...	...	...	...	...	
2496	12.198566	12.020404	0.168752	0.414101	5.932631e-02	-0.137137	0.015511	
2497	12.196315	12.018382	0.168662	0.414094	5.929201e-02	-0.137226	0.015503	

6.1.6.15 Renaming df_hist_em

In [53]:
```
df_hist_em = df_hist_em_raw.copy()

# I only want emissions data from the "Fuel Combustion" section of the Nationa
# Greenhouse Inventory
df_hist_em = df_hist_em.loc[
    (df_hist_em['category (IPCC2006_PRIMAP)'] == '1.A') |
    (df_hist_em['category (IPCC2006_PRIMAP)'] == '0')]

# HISTCR: In this scenario country-reported data (CRF, BUR, UNFCCCDI)
# HISTTP: In this scenario third-party data (CDIAC, FAO, Andrew, EDGAR, BP)
# I'll use third-party data
df_hist_em = df_hist_em.loc[df_hist_em[
    'scenario (PRIMAP-hist)'] == 'HISTTP']

# I'm only concerned with CH4 and CO2
df_hist_em = df_hist_em.loc[
    (df_hist_em['entity'] == 'CH4') |
    (df_hist_em['entity'] == 'CO2')]

# I'll create a list of the columns for the transposed DataFrame

col_idx = []
country_df_hist_em = []

for i in df_hist_em.index:
    country = df_hist_em.loc[i,'area (ISO3)']
    if country == 'EARTH':
        country = 'Earth'
    if country in ['ANNEXI', 'AOSIS', 'BASIC', 'EU27BX',
        'LDC', 'NONANNEXI', 'UMBRELLA', 'OWID_USS']:
        df_hist_em.drop(index=i, inplace=True)
        continue

    sector = df_hist_em.loc[i, 'category (IPCC2006_PRIMAP)']
    if sector == '0':
        sector = '(tot)'
    else:
        sector = '(fuel)'
    gas = df_hist_em.loc[i, 'entity']
    new_index = country + ' ' + gas + ' ' + sector +' (Gg/yr)'
    col_idx.append(new_index)

# I don't need the early columns anymore.
df_hist_em = df_hist_em.iloc[:,6:]

df_hist_em = df_hist_em.transpose()
df_hist_em.columns = col_idx
df_hist_em = df_hist_em
df_hist_em.index = df_hist_em.index.astype(int)
df_hist_em.index.name = 'Year'
```

```
df_hist_em
```

Out[53]:

Year	ABW CH4 (tot) (Gg/yr)	ABW CH4 (fuel) (Gg/yr)	ABW CO2 (tot) (Gg/yr)	ABW CO2 (fuel) (Gg/yr)	AFG CH4 (tot) (Gg/yr)	AFG CH4 (fuel) (Gg/yr)	AFG CO2 (tot) (Gg/yr)	AFG CO (fuel) (Gg/yr)
1750	0.012971	0.005636	0.00000	0.00000	84.999231	1.927227	2.775558e-15	0.00(
1751	0.013068	0.005672	0.00000	0.00000	85.353220	1.935082	2.656374e-01	0.00(
1752	0.013164	0.005708	0.00000	0.00000	85.707207	1.942935	5.312749e-01	0.00(
1753	0.013260	0.005744	0.00000	0.00000	86.061192	1.950788	7.969123e-01	0.00(
1754	0.013356	0.005780	0.00000	0.00000	86.415176	1.958638	1.062550e+00	0.00(

In [54]:
```
hasna = []
hasna_num = []
idx_has_na = []
col_num = -1
for col in df_hist_em:
    col_num+=1
    if np.sum(np.sum(df_hist_em.loc[:,col].isna())) > 0:
        print(col)
        if col not in hasna:
            hasna.append(col)
            hasna_num.append(col_num)
        for idx in df_hist_em.index:
            idx_num = idx - 1750
            tst = df_hist_em.iloc[idx_num:idx_num+1,col_num].isna().values[0]
            if tst:
                if idx not in idx_has_na:
                    idx_has_na.append(idx)
hasna
```

```
AIA CO2 (fuel) (Gg/yr)
AND CO2 (fuel) (Gg/yr)
ATA CO2 (tot) (Gg/yr)
ATA CO2 (fuel) (Gg/yr)
GBR CO2 (fuel) (Gg/yr)
MCO CO2 (fuel) (Gg/yr)
NRU CO2 (fuel) (Gg/yr)
SHN CO2 (fuel) (Gg/yr)
SMR CO2 (fuel) (Gg/yr)
TUV CO2 (fuel) (Gg/yr)
VAT CO2 (fuel) (Gg/yr)
```

Out[54]:
```
['AIA CO2 (fuel) (Gg/yr)',
 'AND CO2 (fuel) (Gg/yr)',
 'ATA CO2 (tot) (Gg/yr)',
 'ATA CO2 (fuel) (Gg/yr)',
 'GBR CO2 (fuel) (Gg/yr)',
 'MCO CO2 (fuel) (Gg/yr)',
 'NRU CO2 (fuel) (Gg/yr)',
 'SHN CO2 (fuel) (Gg/yr)',
 'SMR CO2 (fuel) (Gg/yr)',
 'TUV CO2 (fuel) (Gg/yr)',
 'VAT CO2 (fuel) (Gg/yr)']
```

In [55]:
```
pd.set_option('display.max_rows',None)
df_hist_em[hasna]
```

Out[55]:

Year	AIA CO2 (fuel) (Gg/yr)	AND CO2 (fuel) (Gg/yr)	ATA CO2 (tot) (Gg/yr)	ATA CO2 (fuel) (Gg/yr)	GBR CO2 (fuel) (Gg/yr)	MCO CO2 (fuel) (Gg/yr)	NRU CO2 (fuel) (Gg/yr)	SHI (fue (Gg
1750	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1751	NaN	NaN	NaN	NaN	9357.4184	NaN	NaN	
1752	NaN	NaN	NaN	NaN	9361.0851	NaN	NaN	
1753	NaN	NaN	NaN	NaN	9361.0851	NaN	NaN	
1754	NaN	NaN	NaN	NaN	9364.7518	NaN	NaN	
1755	NaN	NaN	NaN	NaN	9368.4185	NaN	NaN	
1756	NaN	NaN	NaN	NaN	10013.7580	NaN	NaN	
1757	NaN	NaN	NaN	NaN	10017.4240	NaN	NaN	

The Emissions world column ignores the fact that these values are missing, so I will do the same. However, I do want to verify that the world column is a sum of the values I think they are.

In [56]:
```
ch4tot = []
ch4fuel = []
co2tot = []
co2fuel = []

for col in df_hist_em.columns:
    if 'CH4 (tot)' in col:
        ch4tot.append(col)
    elif 'CH4 (fuel)' in col:
        ch4fuel.append(col)
    elif 'CO2 (tot)' in col:
        co2tot.append(col)
    elif 'CO2 (fuel)' in col:
        co2fuel.append(col)
df_hist_em_co2tot = df_hist_em.loc[:,co2fuel]
sum_test = df_hist_em_co2tot.drop(columns='Earth CO2 (fuel) (Gg/yr)').copy()

for idx in sum_test.index:
    sum_test.loc[idx,'sum'] = round(np.sum(sum_test.loc[idx,:]),-2)
sum_test['World'] = df_hist_em_co2tot['Earth CO2 (fuel) (Gg/yr)']
print(np.sum(sum_test.iloc[:,-2] == round(sum_test.iloc[:,-1],-2)))
```

```
272
```

This proves that the world column is the sum of all the other columns within ten gigatons. That's good enough for me.

Unfortunately, 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.

Create Lists of Each Region As Defined by the IEA

North America

In [57]:
```
n_america = ['CAN', 'MEX', 'USA']
```

Central & South America

In [58]:
```
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 [59]:
```
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 [60]:
```
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 [61]:
```
n_africa = ['DZA', 'EGY', 'ESH', 'LBY', 'MAR', 'TUN']
```

In [62]:
```
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 [63]:
```
africa = n_africa + sub_africa
```

Middle East

In [64]:
```
mid_east = ['BHR', 'IRQ', 'IRN', 'JOR', 'KWT', 'LBN', 'OMN', 'QAT', 'SAU',
            'SYR', 'ARE', 'YEM']
```

Eurasia

In [65]:
```
caspian = ['ARM', 'AZE', 'GEO', 'KAZ', 'KGZ', 'TJK', 'TKM', 'UZB']
```

In [66]:
```
eurasia = caspian + ['RUS']
```

Asia Pacific

In [67]:
```
se_asia = ['BRN', 'KHM', 'IDN', 'LAO', 'MYS', 'MMR', 'PHL', 'SGP', 'THA', 'VNM
```

In [68]:
```
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 [69]:
```
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 [70]:
```
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 [71]:
```
opec = ['DZA', 'AGO', 'COG', 'GNQ', 'GAB', 'IRN', 'IRQ', 'KWT', 'LBY', 'NGA',
        'SAU', 'ARE', 'VEN']
```

In [72]:
```
adv_eco = oecd + ['BGR', 'HRV', 'CYP', 'MLT', 'ROU']
```

In [73]:
```
world = (n_america + cen_s_america + europe + africa + mid_east + eurasia +
         asia_pacific)

world.sort()
```

In [74]:
```
dev_asia = asia_pacific.copy()
dev_asia.remove('AUS')
dev_asia.remove('JPN')
dev_asia.remove('KOR')
dev_asia.remove('NZL')
```

In [75]:
```
# 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 [76]:
```
l_america = cen_s_america + ['MEX']
```

In [77]:
```
# 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 [78]:
```
# 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']
```

In [79]:
```
df_hist_em_reg = df_hist_em.copy()
df_hist_em_zero = df_hist_em.copy()
df_hist_em_zero.replace(np.nan,0, inplace = True)

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

col_type = [' CO2 (tot) (Gg/yr)', ' CO2 (fuel) (Gg/yr)',
            ' CH4 (tot) (Gg/yr)', ' CH4 (fuel) (Gg/yr)']

reg_col = []
world_col = []
renaming = []

for c in col_type:
    reg_col.append('Earth' + c)
    world_col.append('Earth' + c)

reg = ['Asia Pacific', 'North America', 'Europe', 'Eurasia',
       'Central & South America', 'Middle East', 'Africa', 'USA', 'CHN',
       'European Union', 'JPN', 'RUS', 'IND',
       'SouthEast Asia', 'BRA']

for r in reg:
    for c in col_type:
        reg_col.append(r+c)

df_hist_em_reg = pd.DataFrame(data = 0,
                              columns = reg_col, index = df_hist_em.index)

df_hist_em_reg[world_col] = df_hist_em[world_col]

for col in df_hist_em_reg.columns:
    if col in df_hist_em.columns:
        df_hist_em_reg[col] = df_hist_em[col]


missing_coun = []
for coun in world:
    for t in col_type:
        tst = coun + t
        r = '?'
        try:
            col = df_hist_em[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 r == '?':
            print("No data for", coun)
            missing_coun.append(coun)
            continue
        df_hist_em_reg[r] = df_hist_em_reg[r] + df_hist_em_zero[tst]

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

df_hist_em_reg = rename_coun(df_hist_em_reg)
df_hist_em_reg
```

Out[79]:

Year	Earth CO2 (tot) (Gg/yr)	Earth CO2 (fuel) (Gg/yr)	Earth CH4 (tot) (Gg/yr)	Earth CH4 (fuel) (Gg/yr)	Asia Pacific CO2 (tot) (Gg/yr)	Asia Pacific CO2 (fuel) (Gg/yr)	Asia Pac CH4 (tot (Gg/yr)
1750	3.393499e+02	1.433389e-01	26929.204	5398.4297	2.885237e-11	0.000000e+00	13430.9
1751	2.974030e+04	9.357564e+03	27003.076	5412.5472	4.181915e+03	0.000000e+00	13463.2
1752	4.978762e+04	9.361233e+03	27076.927	5426.6235	8.363829e+03	0.000000e+00	13495.4
1753	6.983115e+04	9.361236e+03	27150.708	5440.6497	1.254574e+04	0.000000e+00	13527.7
1754	8.987847e+04	9.364905e+03	27224.467	5454.6348	1.672766e+04	0.000000e+00	13560.0
...	...	...	...	...	...	...	
2017	3.672888e+07	3.216032e+07	350975.660	14088.7800	1.547372e+07	1.418122e+07	113104.6
2018	3.775122e+07	3.289954e+07	356107.500	13644.1680	1.592764e+07	1.453714e+07	114479.8
2019	3.800676e+07	3.285008e+07	359266.870	13407.7430	1.621020e+07	1.471351e+07	115106.6
2020	3.578528e+07	3.092440e+07	361919.030	13219.3570	1.596967e+07	1.443630e+07	118303.9
2021	3.775629e+07	3.270272e+07	365021.310	13432.3140	1.686781e+07	1.530250e+07	119156.8

272 rows × 64 columns

6.1.6.16 Renaming df_proj_em_co2

In [80]:
```
df_proj_em_co2 = df_proj_em_co2_raw.copy()

pd.set_option('display.max_rows',100)

# For starters, this is in Gt/yr, while the other emissions data is in Gg/yr.

df_proj_em_co2.iloc[:, 1:] = df_proj_em_co2.iloc[:, 1:] * 1000000

df_proj_em_co2.columns = ['Year', 'Earth - CO2 SSP3-7.0 (tot) (Gg/yr)',
                                  'Earth - CO2 SSP1-1.9 (tot) (Gg/yr)',
                                  'Earth - CO2 SSP1-2.6 (tot) (Gg/yr)',
                                  'Earth - CO2 SSP2-4.5 (tot) (Gg/yr)',
                                  'Earth - CO2 SSP3-8.5 (tot) (Gg/yr)']

df_proj_em_co2 = df_proj_em_co2.iloc[:,[0,2,3,4,1,5]]
df_proj_em_co2
```

Out[80]:

	Year	Earth - CO2 SSP1-1.9 (tot) (Gg/yr)	Earth - CO2 SSP1-2.6 (tot) (Gg/yr)	Earth - CO2 SSP2-4.5 (tot) (Gg/yr)	Earth - CO2 SSP3-7.0 (tot) (Gg/yr)	Earth - CO2 SSP3-8.5 (tot) (Gg/yr)
0	2015	3.915273e+07	3.915273e+07	3.915273e+07	3.915273e+07	3.915273e+07
1	2020	3.969373e+07	3.980401e+07	4.064753e+07	4.480804e+07	4.371235e+07
2	2030	2.284727e+07	3.473442e+07	4.347606e+07	5.284736e+07	5.529658e+07
3	2040	1.047509e+07	2.650918e+07	4.425290e+07	5.849797e+07	6.877570e+07
4	2050	2.050362e+06	1.796354e+07	4.346219e+07	6.290406e+07	8.329822e+07
5	2060	-1.525978e+06	1.052798e+07	4.019648e+07	6.656837e+07	1.003386e+08
6	2070	-4.476970e+06	4.476328e+06	3.523543e+07	7.004198e+07	1.168052e+08
7	2080	-7.308783e+06	-3.285043e+06	2.683837e+07	7.340523e+07	1.296470e+08
8	2090	-1.056502e+07	-8.385183e+06	1.632439e+07	7.779905e+07	1.305762e+08
9	2100	-1.388979e+07	-8.617786e+06	9.682859e+06	8.272583e+07	1.262873e+08

6.1.6.17 Renaming df_proj_em_ch4

In [81]:
```
df_proj_em_ch4 = df_proj_em_ch4_raw.copy()

# This is in Mt/yr, while the other emissions data is in Gg/yr.

df_proj_em_ch4.iloc[:, 1:] = df_proj_em_ch4.iloc[:, 1:] * 1000
# Though this says it's in Mt of ch4, the report clarifies that it's actually
# in Mt of CH4.# Units in Gt_ch4/yr

df_proj_em_ch4.columns = ['Year', 'Earth - CH4 SSP3-7.0 (tot) (Gg/yr)',
                                  'Earth - CH4 SSP1-1.9 (tot) (Gg/yr)',
                                  'Earth - CH4 SSP1-2.6 (tot) (Gg/yr)',
                                  'Earth - CH4 SSP2-4.5 (tot) (Gg/yr)',
                                  'Earth - CH4 SSP3-8.5 (tot) (Gg/yr)']

df_proj_em_ch4 = df_proj_em_ch4.iloc[:,[0,2,3,4,1,5]]
df_proj_em_ch4
```

Out[81]:

	Year	Earth - CH4 SSP1-1.9 (tot) (Gg/yr)	Earth - CH4 SSP1-2.6 (tot) (Gg/yr)	Earth - CH4 SSP2-4.5 (tot) (Gg/yr)	Earth - CH4 SSP3-7.0 (tot) (Gg/yr)	Earth - CH4 SSP3-8.5 (tot) (Gg/yr)
0	2015	388072.795663	388072.795663	388072.795663	388072.795663	388072.795663
1	2020	358907.754705	347490.968043	388090.572662	418454.211984	397464.394299
2	2030	243027.020440	285852.108555	399446.245068	471330.793451	454467.560029
3	2040	200612.861931	242790.147759	382796.746230	515545.705002	542292.737080
4	2050	170196.861887	211108.799575	357166.938785	558974.274241	590297.639509
5	2060	157558.897232	190891.541337	329321.436369	602318.529264	597910.178870
6	2070	146459.703506	166436.311845	316639.039679	647129.405500	597128.413114
7	2080	132247.397298	139312.168422	302907.283508	693382.453028	568979.040418
8	2090	120764.306099	130129.686254	298816.324016	735323.192603	513330.816601
9	2100	111721.246090	122195.342903	295152.936610	777732.192287	482145.484923

6.1.6.18 Renaming df_hist_elec

In [82]:
```
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[82]:

	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 from nuclea (TWh)
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 [83]:
```
# I want to create a dictionary to reference the country codes used in the
# electricity production data.

country_code_elec = {}
country_nan_elec = []

for i in df_hist_elec.index:
    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'])
    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 [84]:
```
# Renaming the columns of the electricity genation DataFrame
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 [85]:
```
# I'll create a dataframe to place rearranged electricity data into.
print('The electricity generation data includes the years',
      df_hist_elec['Year'].min(), '-', str(df_hist_elec['Year'].max())+'.')
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 [86]:
```
world_ptot = world + ['OWID_WRL']
world_ptot.sort()

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[86]:

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)	A
1965	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1966	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1967	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1968	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1969	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1970	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1971	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1972	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

In [87]:
```
# 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 [88]:
```
# There are only four values in 2021 that are missing. Let's 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[88]:

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)	G 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)	G 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 to that.

I'll also drop the entire 2022 column. To get these two numbers, I followed trend lines and took ten year averages, depending on the column.took the 10 year average of hydro power in Albania and added .47 TWh due to a power plant opening in mid 2021, then I increased the solar TWh by the same amount increased between 2019 and 2020.

In [89]:
```
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[89]:

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 six countries are missing data. I'll take a look at what's going on.

In [90]:
```
na_2000 = []
na_2001 = []
na_2000_test = df_hist_elec_yr.loc[2000].isna()
na_2001_test = df_hist_elec_yr.loc[2001].isna()

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

for truth in na_2001_test.index:
    if na_2001_test[truth] == True:
        na_2001.append(truth)

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

Out[90]:

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:

- 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: All 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.

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

In [91]:
```
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[2001,na_2001] = [0, 1.79, 1.13, 0, 0.31, 0.06]
df_hist_elec_yr.loc[2002,na_2001] = [0, 1.76, 1.15, 0, 0.32, 0.06]
df_hist_elec_yr.loc[2003,na_2001[:-1]] = [0, 1.83, 1.18, 0, 0.33]
df_hist_elec_yr.loc[2004,na_2001[:-1]] = [0, 1.82, 1.2, 0, 0.34]
df_hist_elec_yr.loc[2005,na_2001[3:5]] = [0, 0.35]
df_hist_elec_yr.loc[2006,na_2001[3:5]] = [0, 0.36]
df_hist_elec_yr.loc[2007,na_2001[3:5]] = [0, 0.37]
df_hist_elec_yr.loc[2008,na_2001[3:5]] = [0, 0.38]
df_hist_elec_yr.loc[2009,na_2001[3:5]] = [0, 0.39]
df_hist_elec_yr.loc[2010,na_2001[3:5]] = [0, 0.4]
df_hist_elec_yr.loc[2011,na_2001[3:5]] = [0, 0.41]

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

Out[91]:

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	1.85	1.10	0.00	0.00	0.00	0.00	0.30	0.06
2001	0.00	1.79	1.13	0.00	0.00	0.03	0.00	0.31	0.06
2002	0.00	1.76	1.15	0.00	0.07	0.07	0.00	0.32	0.06
2003	0.00	1.83	1.18	0.00	0.16	0.16	0.00	0.33	0.06
2004	0.00	1.82	1.20	0.00	0.19	0.19	0.00	0.34	0.07
2005	0.00	1.87	1.00	0.00	0.23	0.23	0.00	0.35	0.07
2006	0.00	1.75	1.20	0.00	0.16	0.16	0.00	0.36	0.07
2007	0.00	1.28	0.86	0.00	0.20	0.20	0.00	0.37	0.09
2008	0.00	1.54	1.29	0.00	0.20	0.20	0.00	0.38	0.11
2009	0.00	2.07	0.69	0.00	0.23	0.23	0.00	0.39	0.13
2010	0.00	2.75	1.27	0.00	0.22	0.22	0.00	0.40	0.14
2011	0.00	1.20	1.45	0.00	0.27	0.27	0.00	0.41	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

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

In [92]:
```
df_hist_elec_yr = df_hist_elec_yr.loc[2000:]
df_hist_elec_yr
```

Out[92]:

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)	ALB Sola (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 [93]:
```
df_hist_elec_yr = df_hist_elec_yr.copy()

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

col_type = [' Other Renewables (TWh)', ' Bioenergy (TWh)', ' Solar (TWh)',
            ' Wind (TWh)', ' Hydro (TWh)', ' Nuclear (TWh)', ' Oil (TWh)',
            ' Gas (TWh)', ' Coal (TWh)']

reg_col = []
world_col = []
rename = []
for c in col_type:
    reg_col.append('OWID_WRL' + c)
    world_col.append('OWID_WRL' + c)
    rename.append('Earth' + c)
renaming = dict(zip(world_col, rename))

for r in reg:
    for c in col_type:
        reg_col.append(r+c)

df_hist_elec_yr_reg = pd.DataFrame(data = 0, columns = reg_col,
                                   index = df_hist_elec_yr.index)

df_hist_elec_yr_reg[world_col] = df_hist_elec_yr[world_col]

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]


missing_coun = []
for coun in world:
    for t in col_type:
        tst = coun + t
        r = '?'
        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 r == '?':
            print("No data for", coun)
            missing_coun.append(coun)
            continue
        df_hist_elec_yr_reg[r] = df_hist_elec_yr_reg[r] + df_hist_elec_yr[tst]

df_hist_elec_yr_reg.rename(columns = renaming, inplace = True)
df_hist_elec_yr_reg = rename_coun(df_hist_elec_yr_reg)
df_hist_elec_yr_reg
```

Out[93]:

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 O R (T
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

6.1.6.19 Renaming df_hist_dem

In [94]:
```
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[94]:
```
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 [95]:
```
# For reasons I'll explain in a later section, I will define which countries
# are in which deivision of the earth.

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

In [96]:
```
# The country info I need is in Code, not Entity
df_hist_dem.drop(columns = 'Entity', inplace = True)
df_hist_dem.sort_values(by = ['Code'], inplace = True)
```

In [97]:
```
# Now, I'll transpose the columns so I can use it with the others
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[97]:

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)	AUS Hist Dem (TW
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	25
2019	0.92	5.98	15.40	7.61	129.64	143.85	6.34	0.16	0.34	24
2020	0.88	5.95	16.60	7.59	126.52	143.40	6.44	0.15	0.33	25
2021	0.93	6.20	16.43	NaN	135.60	149.18	6.69	0.16	0.35	24
2022	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

33 rows × 216 columns

In [98]:
```
# 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 [99]:
```
# There are only four values in 2021 that are missing. Let's 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[99]:

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. I'll also drop the entire 2022 column. To get these three numbers, I took the five year average. Also, I checked. These numbers are not a summation of the produced electricity.

In [100]:
```
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[100]:
```
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 [101]:
```
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[101]:

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
...	...	...	...
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

22 rows × 3 columns

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 [102]:
```
na_2000[1]
```

Out[102]:
```
'SSD Historic Demand (TWh)'
```

In [103]:
```
df_hist_dem_yr.loc[2000,na_2000] = [4.62, 0.3, 0.06]
df_hist_dem_yr.loc[2001,na_2000] = [4.63, 0.31, 0.06]
df_hist_dem_yr.loc[2002,na_2000] = [4.63, 0.32, 0.06]
df_hist_dem_yr.loc[2003,na_2000[:-1]] = [4.51, 0.33]
df_hist_dem_yr.loc[2004,na_2000[:-1]] = [4.44, 0.34]
df_hist_dem_yr.loc[2005,na_2000[1]] = 0.35
df_hist_dem_yr.loc[2006,na_2000[1]] = 0.36
df_hist_dem_yr.loc[2007,na_2000[1]] = 0.37
df_hist_dem_yr.loc[2008,na_2000[1]] = 0.38
df_hist_dem_yr.loc[2009,na_2000[1]] = 0.39
df_hist_dem_yr.loc[2010,na_2000[1]] = 0.4
df_hist_dem_yr.loc[2011,na_2000[1]] = 0.41

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

Out[103]:

Year	MNE Historic Demand (TWh)	SSD Historic Demand (TWh)	TLS Historic Demand (TWh)
2000	4.62	0.30	0.06
2001	4.63	0.31	0.06
2002	4.63	0.32	0.06
2003	4.51	0.33	0.06
2004	4.44	0.34	0.07
...	...	...	...
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

22 rows × 3 columns

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

In [104]:
```
df_hist_dem_yr = df_hist_dem_yr.loc[2000:]
df_hist_dem_yr
```

Out[104]:

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)	AUS Hist Dem (TW
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
...	...	...	...	...	...	...	...	...	...	
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

22 rows × 216 columns

In [105]:
```
df_hist_dem_yr = df_hist_dem_yr.copy()

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

col_type = [' Historic Demand (TWh)']

reg_col = []
world_col = []
rename = []
for c in col_type:
    reg_col.append('OWID_WRL' + c)
    world_col.append('OWID_WRL' + c)
    rename.append('Earth' + c)
renaming = dict(zip(world_col, rename))

for r in reg:
    for c in col_type:
        reg_col.append(r+c)

df_hist_dem_yr_reg = pd.DataFrame(data = 0, columns = reg_col,
                                  index = df_hist_dem_yr.index)

df_hist_dem_yr_reg[world_col] = df_hist_dem_yr[world_col]

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]


missing_coun = []
for coun in world:
    for t in col_type:
        tst = coun + t
        r = '?'
        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 r == '?':
```

```
            print("No data for", coun)
            missing_coun.append(coun)
            continue
        df_hist_dem_yr_reg[r] = df_hist_dem_yr_reg[r] + df_hist_dem_yr[tst]

df_hist_dem_yr_reg.rename(columns=renaming,inplace = True)
df_hist_dem_yr_reg = rename_coun(df_hist_dem_yr_reg)
df_hist_dem_yr_reg
```

Out[105]:

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)	Ch His De (TV
2000	14971.90	3753.73	4583.44	1014.49	987.19	778.62	430.47	423.01	3835.86	13
2001	15196.46	3922.35	4500.90	1022.65	996.68	768.05	455.59	438.41	3749.60	14
2002	15733.31	4176.54	4632.77	1031.25	1014.12	792.29	487.66	463.28	3865.21	16
2003	16291.95	4483.46	4643.67	1048.37	1038.29	831.39	517.99	486.39	3875.36	19
2004	17093.28	4921.23	4754.45	1071.52	1065.23	877.23	549.85	515.69	3963.26	21
...	...	...	...	...	...	...	...	...	...	
2017	25026.85	10445.66	5000.66	1195.35	1271.50	1271.14	1109.36	805.64	4108.62	65
2018	26022.34	11158.56	5166.96	1208.64	1288.76	1291.25	1113.83	820.54	4246.01	71
2019	26366.21	11517.82	5097.23	1197.71	1303.09	1293.64	1149.14	827.63	4197.42	74
2020	26275.23	11708.52	4986.52	1178.12	1295.58	1291.84	1142.57	806.57	4090.49	77
2021	27812.74	12667.08	5118.78	1232.56	1364.76	1362.57	1211.87	839.32	4191.53	84

22 rows × 16 columns

6.1.6.20 Renaming df_proj_dem

In [106]:
```
# 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 [107]:
```
# I don't know what the carbon data is in this
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[107]:
```
Stated Policies Scenario              1796
Announced Pledges Scenario             768
Net Zero Emissions by 2050 Scenario     30
Name: Scenario, dtype: int64
```

In [108]:
```
# 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 [109]:
```
# Now I want to see what the Product column does.
df_proj_dem_ann['Product'].value_counts()
```

Out[109]:
```
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 [110]:
```
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[110]:
```
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 [111]:
```
# 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[111]:

	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

Wow. These conversions 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.

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

In [112]:
```
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[112]:

	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 [113]:
```
# 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 [114]:
```
# 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 [115]:
```
df_proj_dem_stat
```

Out[115]:

	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 [116]:
```
# 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 and dropping Southeast Asia
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[116]:

	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)	Africa Projecte Demand (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

In [117]:
```
# A look at the regions included in this Dataset

df_proj_dem['Region'].value_counts()
```

Out[117]:
```
World             481
Africa            149
Southeast Asia    149
North America     149
Middle East       149
                 ...
Other              10
Atlantic Basin     10
East of Suez       10
Non-OPEC            7
OPEC                7
Name: Region, Length: 22, dtype: int64
```

6.1.6.21 Renaming df_ssp_proj

Some explanation is required before I begin cleaning this dataset.

Shared Socio-Economic Pathways (SSP) (Five scenarios and four variants)

The Elmar Kriegler and colleagues defined five different Socio-Economic Pathways (SSP) for the world to follow into the future. Each SSP scenario (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_Chapter01.pdf#page=7) shares different possible future developments of anthropogenic drivers of climate change. They cover a broad range of emissions pathways, including new low-emissions pathways. The eight SSP scenarios I will use in this analysis are:

SSP1 Sustainability – Taking the Green Road (Low challenges to mitigation and adaptation)
The world shifts gradually, but pervasively, toward a more sustainable path, emphasizing more inclusive development that respects perceived environmental boundaries. Management of the global commons slowly improves, educational and health investments accelerate the demographic transition, and the emphasis on economic growth shifts toward a broader emphasis on human well-being. Driven by an increasing commitment to achieving development goals, inequality is reduced both across and within countries. Consumption is oriented toward low material growth and lower resource and energy intensity.

SSP2 Middle of the Road (Medium challenges to mitigation and adaptation)
The world follows a path in which social, economic, and technological trends do not shift markedly from historical patterns. Development and income growth proceeds unevenly, with some countries making relatively good progress while others fall short of expectations. Global and national institutions work toward but make slow progress in achieving sustainable development goals. Environmental systems experience degradation, although there are some improvements and overall the intensity of resource and energy use declines. Global population growth is moderate and levels off in the second half of the century. Income inequality persists or improves only slowly and challenges to reducing vulnerability to societal and environmental changes remain.

SSP3 Regional Rivalry – A Rocky Road (High challenges to mitigation and adaptation)
A resurgent nationalism, concerns about competitiveness and security, and regional conflicts push countries to increasingly focus on domestic or, at most, regional issues. Policies shift over time to become increasingly oriented toward national and regional security issues. Countries focus on achieving energy and food security goals within their own regions at the expense of broader-based development. Investments in education and technological development decline. Economic development is slow, consumption is material-intensive, and inequalities persist or worsen over time. Population growth is low in industrialized and high in developing countries. A low international priority for addressing environmental concerns leads to strong environmental degradation in some regions.

SSP4 Inequality – A Road Divided (Low challenges to mitigation, high challenges to adaptation)
Highly unequal investments in human capital, combined with increasing disparities in economic opportunity and political power, lead to increasing inequalities and stratification both across and within countries. Over time, a gap widens between an internationally-connected society that contributes to knowledge- and capital-intensive sectors of the global economy, and a fragmented collection of lower-income, poorl

```
y educated societies that work in a labor intensive, low-tech econom
y. Social cohesion degrades and conflict and unrest become increasing
ly common. Technology development is high in the high-tech economy an
d sectors. The globally connected energy sector diversifies, with inv
estments in both carbon-intensive fuels like coal and unconventional
oil, but also low-carbon energy sources. Environmental policies focus
on local issues around middle and high income areas.

**SSP5** Fossil-fueled Development – Taking the Highway (High challen
ges to mitigation, low challenges to adaptation)
This world places increasing faith in competitive markets, innovation
and participatory societies to produce rapid technological progress a
nd development of human capital as the path to sustainable developmen
t. Global markets are increasingly integrated. There are also strong
investments in health, education, and institutions to enhance human a
nd social capital. At the same time, the push for economic and social
development is coupled with the exploitation of abundant fossil fuel
resources and the adoption of resource and energy intensive lifestyle
s around the world. All these factors lead to rapid growth of the glo
bal economy, while global population peaks and declines in the 21st c
entury. Local environmental problems like air pollution are successfu
lly managed. There is faith in the ability to effectively manage soci
al and ecological systems, including by geo-engineering if necessary.
```

Further, the IPCC selected (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_Chapter01.pdf#page=89) nine SSP scenarios for further study. Like the article above, each SSP scenario (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_Chapter01.pdf#page=7) defines ways in which mankind can guide the future, but with variants of the five models included. The models are defined as SSPX-Y, with X being the SSP scenario number, and Y indicates the approximate radiative forcing value reached by 2100 (https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_Chapter01.pdf#page=87) Below, I've listed the SSPs I'll look at in this project, copied from the IPCC report:

```
**SSP1-1.9 (Sustainability)** Holds warming to approximately 1.5°C ab
ove 1850–1900 in 2100 after slight overshoot (median) and implied net
zero CO2 emissions around the middle of the century.

**SSP1-2.6 (Sustainability, later net zero)** Stays below 2.0°C warmi
ng relative to 1850–1900 (median) with implied net zero CO2 emissions
in the second half of the century.

**SSP4-3.4 (Inequality w/ Climate Policy)** A scenario between SSP1-
2.6 and SSP2-4.5 in terms of end-of-century radiative forcing. It doe
s not stay below 2.0°C in most CMIP6 runs (Chapter 4) relative to 185
0–1900.
```

Primary, Secondary and Final Energy (https://www.ipcc.ch/site/assets/uploads/2018/02/ipcc_wg3_ar5_chapter7.pdf#page=9)

International Institute For Applied System Analysis has a public database (https://tntcat.iiasa.ac.at/SspDb/dsd?Action=htmlpage&page=10) with modeling from five different groups that target the same SSPs. However, after some study I realized that, despite having similar titles, the effective radiative forcing they targeted with these public models are overall warming, including all anthropogenic factors, at 2100 where the IPCC defines the Y-value in the scenario number as the radiative forcing of CO2 alone. However, I may be able to glean some information from this.

In [118]:
```
pd.set_option("display.max_rows", 150)

df_ssp_proj = df_ssp_proj_raw.copy()

# The MODEL column is perfectly matched by the SSP. It represents differe
# teams creating the model for different SSP scenarios. It's not needed.

#df_ssp_proj.drop(columns = 'MODEL', inplace = True)

renaming = {'MODEL' : 'Model',
            'SCENARIO' : 'Scenario',
            'REGION' : 'Region',
            'VARIABLE' : 'Variable',
            'UNIT' : 'Unit'}

df_ssp_proj.rename(columns = renaming, inplace = True)

# I want to see what data is available in the Variables category
df_ssp_proj['Variable'].unique()
```

Out[118]:
```
array(['Agricultural Demand|Crops', 'Agricultural Demand|Crops|Energy',
       'Agricultural Demand|Livestock',
       'Agricultural Production|Crops|Energy',
       'Agricultural Production|Crops|Non-Energy',
       'Agricultural Production|Livestock', 'Capacity|Electricity',
       'Capacity|Electricity|Biomass', 'Capacity|Electricity|Coal',
       'Capacity|Electricity|Gas', 'Capacity|Electricity|Geothermal',
       'Capacity|Electricity|Hydro', 'Capacity|Electricity|Nuclear',
       'Capacity|Electricity|Oil', 'Capacity|Electricity|Solar',
       'Capacity|Electricity|Solar|PV', 'Capacity|Electricity|Wind',
       'Capacity|Electricity|Wind|Onshore', 'Consumption', 'Emissions|BC',
       'Emissions|CH4', 'Emissions|CH4|Land Use', 'Emissions|CO',
       'Emissions|CO2', 'Emissions|CO2|Carbon Capture and Storage',
       'Emissions|CO2|Carbon Capture and Storage|Biomass',
       'Emissions|CO2|Fossil Fuels and Industry',
       'Emissions|CO2|Land Use', 'Emissions|F-Gases',
       'Emissions|Kyoto Gases', 'Emissions|N2O', 'Emissions|N2O|Land Use',
       'Emissions|NH3', 'Emissions|NOx', 'Emissions|OC',
       'Emissions|Sulfur', 'Emissions|VOC', 'Final Energy',
       'Final Energy|Electricity', 'Final Energy|Gases',
       'Final Energy|Heat', 'Final Energy|Industry',
       'Final Energy|Liquids', 'Final Energy|Residential and Commercial',
       'Final Energy|Solids', 'Final Energy|Solids|Biomass',
       'Final Energy|Solids|Coal', 'Final Energy|Transportation',
       'GDP|PPP', 'Land Cover|Built-up Area', 'Land Cover|Cropland',
       'Land Cover|Forest', 'Land Cover|Pasture', 'Population',
       'Price|Carbon', 'Primary Energy', 'Primary Energy|Biomass',
       'Primary Energy|Biomass|w/ CCS', 'Primary Energy|Biomass|w/o CCS',
       'Primary Energy|Coal', 'Primary Energy|Coal|w/ CCS',
       'Primary Energy|Coal|w/o CCS', 'Primary Energy|Fossil',
       'Primary Energy|Fossil|w/ CCS', 'Primary Energy|Fossil|w/o CCS',
       'Primary Energy|Gas', 'Primary Energy|Gas|w/ CCS',
       'Primary Energy|Gas|w/o CCS', 'Primary Energy|Geothermal',
       'Primary Energy|Hydro', 'Primary Energy|Non-Biomass Renewables',
       'Primary Energy|Nuclear', 'Primary Energy|Oil',
       'Primary Energy|Oil|w/ CCS', 'Primary Energy|Oil|w/o CCS',
       'Primary Energy|Secondary Energy Trade', 'Primary Energy|Solar',
       'Primary Energy|Wind', 'Secondary Energy|Electricity',
       'Secondary Energy|Electricity|Biomass',
       'Secondary Energy|Electricity|Biomass|w/ CCS',
       'Secondary Energy|Electricity|Biomass|w/o CCS',
       'Secondary Energy|Electricity|Coal',
       'Secondary Energy|Electricity|Coal|w/ CCS',
       'Secondary Energy|Electricity|Coal|w/o CCS',
       'Secondary Energy|Electricity|Gas',
       'Secondary Energy|Electricity|Gas|w/ CCS',
       'Secondary Energy|Electricity|Gas|w/o CCS',
       'Secondary Energy|Electricity|Geothermal',
       'Secondary Energy|Electricity|Hydro',
       'Secondary Energy|Electricity|Non-Biomass Renewables',
       'Secondary Energy|Electricity|Nuclear',
       'Secondary Energy|Electricity|Oil',
       'Secondary Energy|Electricity|Solar',
       'Secondary Energy|Electricity|Wind', 'Secondary Energy|Gases',
       'Secondary Energy|Gases|Natural Gas', 'Secondary Energy|Heat',
       'Secondary Energy|Liquids', 'Secondary Energy|Liquids|Biomass',
       'Secondary Energy|Liquids|Oil', 'Primary Energy|Other',
```

```
       'Secondary Energy|Heat|Geothermal',
       'Diagnostics|MAGICC6|Concentration|CH4',
       'Diagnostics|MAGICC6|Concentration|CO2',
       'Diagnostics|MAGICC6|Concentration|N2O',
       'Diagnostics|MAGICC6|Forcing',
       'Diagnostics|MAGICC6|Forcing|Aerosol',
       'Diagnostics|MAGICC6|Forcing|CH4',
       'Diagnostics|MAGICC6|Forcing|CO2',
       'Diagnostics|MAGICC6|Forcing|F-Gases',
       'Diagnostics|MAGICC6|Forcing|Kyoto Gases',
       'Diagnostics|MAGICC6|Forcing|N2O',
       'Diagnostics|MAGICC6|Temperature|Global Mean',
       'Harmonized Emissions|BC',
       'Harmonized Emissions|CH4|Fossil Fuels and Industry',
       'Harmonized Emissions|CH4|Land Use', 'Harmonized Emissions|CO',
       'Harmonized Emissions|CO2|Fossil Fuels and Industry',
       'Harmonized Emissions|CO2|Land Use', 'Harmonized Emissions|NH3',
       'Harmonized Emissions|NOx', 'Harmonized Emissions|OC',
       'Harmonized Emissions|Sulfur', 'Harmonized Emissions|F-Gases',
       'Harmonized Emissions|Kyoto Gases', 'Harmonized Emissions|VOC',
       'Capacity|Electricity|Other',
       'Emissions|CH4|Fossil Fuels and Industry',
       'Energy Service|Transportation|Freight',
       'Energy Service|Transportation|Passenger', 'Final Energy|Hydrogen',
       'Final Energy|Solids|Biomass|Traditional',
       'Primary Energy|Biomass|Traditional',
       'Secondary Energy|Gases|Biomass', 'Secondary Energy|Gases|Coal',
       'Secondary Energy|Hydrogen', 'Secondary Energy|Hydrogen|Biomass',
       'Secondary Energy|Hydrogen|Biomass|w/ CCS',
       'Secondary Energy|Hydrogen|Biomass|w/o CCS',
       'Secondary Energy|Hydrogen|Electricity',
       'Secondary Energy|Liquids|Biomass|w/ CCS',
       'Secondary Energy|Liquids|Biomass|w/o CCS',
       'Secondary Energy|Liquids|Coal',
       'Secondary Energy|Liquids|Coal|w/ CCS',
       'Secondary Energy|Liquids|Coal|w/o CCS',
       'Secondary Energy|Liquids|Gas',
       'Secondary Energy|Liquids|Gas|w/ CCS',
       'Secondary Energy|Liquids|Gas|w/o CCS', 'Secondary Energy|Solids',
       'Capacity|Electricity|Solar|CSP',
       'Capacity|Electricity|Wind|Offshore', 'Final Energy|Solar'],
      dtype=object)
```

In [119]:
```
# Now I want to see what SSPs these models are using.
df_ssp_proj['Scenario'].unique()
```

Out[119]:
```
array(['SSP1-19', 'SSP1-26', 'SSP1-34', 'SSP1-45', 'SSP1-Baseline',
       'SSP2-19', 'SSP2-26', 'SSP2-34', 'SSP2-45', 'SSP2-60',
       'SSP2-Baseline', 'SSP3-34', 'SSP3-45', 'SSP3-60', 'SSP3-Baseline',
       'SSP4-26', 'SSP4-34', 'SSP4-45', 'SSP4-Baseline', 'SSP5-26',
       'SSP5-34', 'SSP5-45', 'SSP5-60', 'SSP5-Baseline', 'SSP4-60',
       'SSP5-19', 'SSP1-60', 'SSP4-19'], dtype=object)
```

In [120]:
```
# Function to make creating dataframes with only specific values out of
# df_ssp_proj simple.
def ssp_trim(col, vals, df):
    temp_idx = []
    for idx in df.index:
        for val in vals:
            if df.loc[idx,col] == val:
                temp_idx.append(idx)
    return df.loc[temp_idx]
```

In [121]:
```
# This model has only eight SSPs that match the IPCC, but it has more SSPs
# overall. I'll cut out the SSPs I don't want to use.

ipcc_ssp = ['SSP1-19', 'SSP1-26', 'SSP4-34', 'SSP2-45', 'SSP4-Baseline',
            'SSP3-Baseline', 'SSP3-45', 'SSP5-Baseline']

df_ssp_proj = ssp_trim('Scenario', ipcc_ssp, df_ssp_proj)

df_ssp_proj[['Model', 'Scenario', 'Region', 'Variable', 'Unit', '2100']].loc[
    (df_ssp_proj['Variable'] == 'Diagnostics|MAGICC6|Forcing') &
    (df_ssp_proj['Region'] == 'World')].sort_values('Scenario')
```

Out[121]:

	Model	Scenario	Region	Variable	Unit	2100
526	AIM/CGE	SSP1-19	World	Diagnostics|MAGICC6|Forcing	W/m2	1.916949
31727	IMAGE	SSP1-19	World	Diagnostics|MAGICC6|Forcing	W/m2	1.905195
58818	REMIND-MAGPIE	SSP1-19	World	Diagnostics|MAGICC6|Forcing	W/m2	1.903088
15101	GCAM4	SSP1-19	World	Diagnostics|MAGICC6|Forcing	W/m2	1.915145
70640	WITCH-GLOBIOM	SSP1-19	World	Diagnostics|MAGICC6|Forcing	W/m2	1.905763
47667	MESSAGE-GLOBIOM	SSP1-19	World	Diagnostics|MAGICC6|Forcing	W/m2	1.850753
71209	WITCH-GLOBIOM	SSP1-26	World	Diagnostics|MAGICC6|Forcing	W/m2	2.622516
15820	GCAM4	SSP1-26	World	Diagnostics|MAGICC6|Forcing	W/m2	2.587939
32478	IMAGE	SSP1-26	World	Diagnostics|MAGICC6|Forcing	W/m2	2.637267
48409	MESSAGE-GLOBIOM	SSP1-26	World	Diagnostics|MAGICC6|Forcing	W/m2	2.610017
1145	AIM/CGE	SSP1-26	World	Diagnostics|MAGICC6|Forcing	W/m2	2.688616
59514	REMIND-MAGPIE	SSP1-26	World	Diagnostics|MAGICC6|Forcing	W/m2	2.658288
53648	MESSAGE-GLOBIOM	SSP2-45	World	Diagnostics|MAGICC6|Forcing	W/m2	4.280675
36619	IMAGE	SSP2-45	World	Diagnostics|MAGICC6|Forcing	W/m2	4.354471
5373	AIM/CGE	SSP2-45	World	Diagnostics|MAGICC6|Forcing	W/m2	4.297851
64391	REMIND-MAGPIE	SSP2-45	World	Diagnostics|MAGICC6|Forcing	W/m2	4.320497
20853	GCAM4	SSP2-45	World	Diagnostics|MAGICC6|Forcing	W/m2	4.059277
75186	WITCH-GLOBIOM	SSP2-45	World	Diagnostics|MAGICC6|Forcing	W/m2	4.263247
7729	AIM/CGE	SSP3-45	World	Diagnostics|MAGICC6|Forcing	W/m2	4.293275
39371	IMAGE	SSP3-45	World	Diagnostics|MAGICC6|Forcing	W/m2	4.263000
56629	MESSAGE-GLOBIOM	SSP3-45	World	Diagnostics|MAGICC6|Forcing	W/m2	4.168694
77456	WITCH-GLOBIOM	SSP3-45	World	Diagnostics|MAGICC6|Forcing	W/m2	4.278123
58113	MESSAGE-GLOBIOM	SSP3-Baseline	World	Diagnostics|MAGICC6|Forcing	W/m2	8.121157
78589	WITCH-GLOBIOM	SSP3-Baseline	World	Diagnostics|MAGICC6|Forcing	W/m2	7.402108
40747	IMAGE	SSP3-Baseline	World	Diagnostics|MAGICC6|Forcing	W/m2	6.901457
23010	GCAM4	SSP3-Baseline	World	Diagnostics|MAGICC6|Forcing	W/m2	7.175920
8961	AIM/CGE	SSP3-Baseline	World	Diagnostics|MAGICC6|Forcing	W/m2	7.165433
42123	IMAGE	SSP4-34	World	Diagnostics|MAGICC6|Forcing	W/m2	3.385370
24498	GCAM4	SSP4-34	World	Diagnostics|MAGICC6|Forcing	W/m2	3.425418
10201	AIM/CGE	SSP4-34	World	Diagnostics|MAGICC6|Forcing	W/m2	3.567347
80295	WITCH-GLOBIOM	SSP4-34	World	Diagnostics|MAGICC6|Forcing	W/m2	3.392989

	Model	Scenario	Region	Variable	Unit	2100
44187	IMAGE	SSP4-Baseline	World	Diagnostics|MAGICC6|Forcing	W/m2	5.729804
81997	WITCH-GLOBIOM	SSP4-Baseline	World	Diagnostics|MAGICC6|Forcing	W/m2	5.902009
26672	GCAM4	SSP4-Baseline	World	Diagnostics|MAGICC6|Forcing	W/m2	6.425488
11383	AIM/CGE	SSP4-Baseline	World	Diagnostics|MAGICC6|Forcing	W/m2	5.894492
30979	GCAM4	SSP5-Baseline	World	Diagnostics|MAGICC6|Forcing	W/m2	7.823262
70039	REMIND-MAGPIE	SSP5-Baseline	World	Diagnostics|MAGICC6|Forcing	W/m2	8.697770
14411	AIM/CGE	SSP5-Baseline	World	Diagnostics|MAGICC6|Forcing	W/m2	8.204301
46939	IMAGE	SSP5-Baseline	World	Diagnostics|MAGICC6|Forcing	W/m2	8.424180
84267	WITCH-GLOBIOM	SSP5-Baseline	World	Diagnostics|MAGICC6|Forcing	W/m2	8.850396

This shows that there are five different teams that produced five different data sets, all attempting to do the same thing and getting slightly different results. For this set only, I'll average all their results together.

In [122]:
```
def ssp_avg():
    jnk = pd.DataFrame(columns = df_ssp_proj.columns)
    jnk.drop(columns = 'Model', inplace = True)
    years = ['2005', '2010', '2020', '2030', '2040', '2050',
             '2060', '2070', '2080', '2090', '2100']
    i = 0
    for scen in df_ssp_proj['Scenario'].unique():
        for reg in df_ssp_proj['Region'].unique():
            for var in df_ssp_proj['Variable'].unique():
                idx = df_ssp_proj.loc[
                        (df_ssp_proj['Scenario'] == scen) &
                        (df_ssp_proj['Region'] == reg) &
                        (df_ssp_proj['Variable'] ==  var)].index
                if len(idx) == 0:
                    continue
                yr_avg = [scen, reg,  var]
                i += 1
                for yr in years:
                    if yr == '2005':
                        yr_avg.append(df_ssp_proj.loc[idx[0],'Unit'])
                    with warnings.catch_warnings():
                        warnings.simplefilter("ignore", category=RuntimeWarnin
                        yr_avg.append(np.nanmean(df_ssp_proj.loc[idx, yr]))
                jnk.loc[len(jnk)] = yr_avg
    return jnk

df_ssp_proj_avg = ssp_avg()
df_ssp_proj_avg
```

Out[122]:

	Scenario	Region	Variable	Unit	2005	2010	2020
0	SSP1-19	R5.2ASIA	Agricultural Demand|Crops	million t DM/yr	1242.740519	1407.337374	1589
1	SSP1-19	R5.2ASIA	Agricultural Demand|Crops|Energy	million t DM/yr	0.000000	85.533243	201
2	SSP1-19	R5.2ASIA	Agricultural Demand|Livestock	million t DM/yr	80.366557	88.180026	110
3	SSP1-19	R5.2ASIA	Agricultural Production|Crops|Energy	million t DM/yr	1.121395	0.849346	82
4	SSP1-19	R5.2ASIA	Agricultural Production|Crops|Non-Energy	million t DM/yr	1348.293099	1465.985921	1638
...	...	...	...	...	...	...	
6609	SSP5-Baseline	World	Secondary Energy|Liquids|Gas|w/o CCS	EJ/yr	NaN	0.211797	5
6610	SSP5-Baseline	World	Secondary Energy|Solids	EJ/yr	59.439266	67.129634	72
6611	SSP5-Baseline	World	Capacity|Electricity|Solar|CSP	GW	0.494484	1.808970	9
6612	SSP5-Baseline	World	Capacity|Electricity|Wind|Offshore	GW	0.868600	3.622350	14
6613	SSP5-Baseline	World	Final Energy|Solar	EJ/yr	0.019956	0.166035	1

6614 rows × 15 columns

In [123]:
```
# Now, I will trim out all of the variables that I don't want.

variables = ['Emissions|CH4', 'Emissions|CO2',
             'Emissions|CO2|Carbon Capture and Storage',
             'Emissions|CO2|Carbon Capture and Storage|Biomass',
             'Emissions|CO2|Fossil Fuels and Industry',
             'Emissions|CO2|Land Use',
             'Secondary Energy|Electricity',
             'Secondary Energy|Electricity|Biomass',
             'Secondary Energy|Electricity|Biomass|w/ CCS',
             'Secondary Energy|Electricity|Biomass|w/o CCS',
             'Secondary Energy|Electricity|Coal',
             'Secondary Energy|Electricity|Coal|w/ CCS',
             'Secondary Energy|Electricity|Coal|w/o CCS',
             'Secondary Energy|Electricity|Gas',
             'Secondary Energy|Electricity|Gas|w/ CCS',
             'Secondary Energy|Electricity|Gas|w/o CCS',
             'Secondary Energy|Electricity|Geothermal',
             'Secondary Energy|Electricity|Hydro',
             'Secondary Energy|Electricity|Non-Biomass Renewables',
             'Secondary Energy|Electricity|Nuclear',
             'Secondary Energy|Electricity|Oil',
             'Secondary Energy|Electricity|Solar',
             'Secondary Energy|Electricity|Wind',
             'Diagnostics|MAGICC6|Concentration|CH4',
             'Diagnostics|MAGICC6|Concentration|CO2',
             'Diagnostics|MAGICC6|Forcing',
             'Diagnostics|MAGICC6|Forcing|Aerosol',
             'Diagnostics|MAGICC6|Forcing|CH4',
             'Diagnostics|MAGICC6|Forcing|CO2',
             'Diagnostics|MAGICC6|Forcing|F-Gases',
             'Diagnostics|MAGICC6|Forcing|Kyoto Gases',
             'Diagnostics|MAGICC6|Forcing|N2O',
             'Diagnostics|MAGICC6|Temperature|Global Mean']

df_ssp_proj_avg = ssp_trim('Variable', variables, df_ssp_proj_avg)
df_ssp_proj_avg
```

Out[123]:

	Scenario	Region	Variable	Unit	2005	2010
20	SSP1-19	R5.2ASIA	Emissions|CH4	Mt CH4/yr	135.265305	148.97
23	SSP1-19	R5.2ASIA	Emissions|CO2	Mt CO2/yr	10894.312138	13371.38
24	SSP1-19	R5.2ASIA	Emissions|CO2|Carbon Capture and Storage	Mt CO2/yr	0.237534	105.75
25	SSP1-19	R5.2ASIA	Emissions|CO2|Carbon Capture and Storage|Biomass	Mt CO2/yr	0.006343	0.00
26	SSP1-19	R5.2ASIA	Emissions|CO2|Fossil Fuels and Industry	Mt CO2/yr	9284.292379	12133.64
...	...	...	...	...	...	
6570	SSP5-Baseline	World	Diagnostics|MAGICC6|Forcing|CO2	W/m2	1.689784	1.83
6571	SSP5-Baseline	World	Diagnostics|MAGICC6|Forcing|F-Gases	W/m2	0.021105	0.03
6572	SSP5-Baseline	World	Diagnostics|MAGICC6|Forcing|Kyoto Gases	W/m2	2.346750	2.53
6573	SSP5-Baseline	World	Diagnostics|MAGICC6|Forcing|N2O	W/m2	0.155711	0.16
6574	SSP5-Baseline	World	Diagnostics|MAGICC6|Temperature|Global Mean	Â°C	0.913386	0.98

1184 rows × 15 columns

In [124]:
```
df_ssp_proj_avg['Region'].value_counts()
```

Out[124]:
```
World       264
R5.2ASIA    184
R5.2LAM     184
R5.2MAF     184
R5.2OECD    184
R5.2REF     184
Name: Region, dtype: int64
```

In [125]:
```
# Now, I'll transpose the columns so I can use it with the others
df_ssp_proj_avg_yr = df_ssp_proj_avg.iloc[:,4:].transpose()
cols = []
for row in df_ssp_proj_avg.index:
    cols.append(df_ssp_proj_avg.loc[row, 'Scenario'] + ' ' +
                df_ssp_proj_avg.loc[row, 'Region'] + ' ' +
                df_ssp_proj_avg.loc[row, 'Variable'] + ' (' +
                df_ssp_proj_avg.loc[row, 'Unit'] + ')')
df_ssp_proj_avg_yr.columns = cols
df_ssp_proj_avg_yr.index = [2005, 2010, 2020, 2030, 2040,
                            2050, 2060, 2070, 2080, 2090, 2100]
df_ssp_proj_avg_yr.index.name = 'Year'
df_ssp_proj_avg_yr
```

Out[125]:

Year	SSP1-19 R5.2ASIA Emissions|CH4 (Mt CH4/yr)	SSP1-19 R5.2ASIA Emissions|CO2 (Mt CO2/yr)	SSP1-19 R5.2ASIA Emissions|CO2|Carbon Capture and Storage (Mt CO2/yr)	SSP1-19 R5.2ASIA Emissions|CO2|Carbon Capture and Storage|Biomass (Mt CO2/yr)	SSP1-19 R5 Emissions| Fuels and I (Mt CO2/yr)
2005	135.265305	10894.312138	0.237534	0.006343	9
2010	148.978877	13371.383419	105.759659	0.003577	12
2020	143.365354	15145.880447	158.214825	6.784811	14
2030	84.643114	8957.265668	727.225398	176.753969	9
2040	69.759692	5248.932496	2057.622996	749.639286	5
2050	57.856241	1942.642955	3555.425560	1523.319801	2
2060	50.294041	-191.269072	4321.855577	2051.058249	
2070	43.211271	-1405.809849	4889.806585	2580.604476	-
2080	36.917238	-2399.958349	4962.475058	3040.061697	-1
2090	31.447407	-3005.693703	4655.551477	3175.339372	-2
2100	26.886229	-3570.824354	4698.256958	3546.469480	-2

6.2 Prepping Data for Modeling and Export

In [126]:
```
pd.set_option("display.max_rows", 6)


# For Effective Radiative Forcers to Global Temperature
# (Before I model this, I will )

df_hist_deg_c_mod = pd.concat([df_deg_c.iloc[1:-1,4:],
                    df_hist_rf.loc[1851:]], axis = 1)

df_hist_deg_c_mod
```

Out[126]:

Year	Global Annual Average Anomaly (°C)	Hist Total (W/m²)	Hist CO2 (W/m²)	Hist CH4 (W/m²)	Hist N2O (W/m²)	Hist Minor GHG (W/m²)	Hist O3 (W/m²)	Hist Stratospheric H2O (W/m²)	His Cor (W/
1851	0.048824	0.340734	0.142307	0.049486	0.007656	8.228277e-08	0.031016	0.004548	0.0
1852	0.078824	0.341728	0.144956	0.049947	0.007954	8.533412e-08	0.032032	0.004591	0.0
1853	0.046324	0.150538	0.147334	0.050708	0.008331	8.839851e-08	0.033049	0.004661	0.0
...	...	...	...	...	...	...	...	...	
2017	1.178824	2.751112	2.088686	0.537665	0.202033	4.046273e-01	0.459667	0.049419	0.0
2018	1.101324	2.783340	2.122080	0.540821	0.205654	4.066049e-01	0.467067	0.049709	0.0
2019	1.226324	2.838193	2.156278	0.543983	0.208463	4.080644e-01	0.474467	0.050000	0.0

169 rows × 17 columns

In [127]:
```
df_hist_deg_c_mod.to_csv(
    "Data/Export/df_hist_deg_c_mod.csv",
    encoding = 'ANSI')
```

In [128]:
```
df_hist_co2_ppm_mod = pd.concat([df_hist_rf.iloc[1:,1],
                                 df_hist_ghg_2014.iloc[1:,0],
                                 df_hist_em.loc[1751:,[
                                     'Earth CO2 (tot) (Gg/yr)',
                                     'Earth CO2 (fuel) (Gg/yr)']]], axis=1)

df_hist_co2_ppm_mod.iloc[-7:-2,1] = df_hist_ghg_2019.iloc[-5:,0]


pd.set_option("display.max_rows", 400)
df_hist_co2_ppm_mod
```

Out[128]:

Year	Hist CO2 (W/m²)	CO2 (ppm)	Earth CO2 (tot) (Gg/yr)	Earth CO2 (fuel) (Gg/yr)
1751	0.001416	277.187988	2.974030e+04	9.357564e+03
1752	0.002832	277.229004	4.978762e+04	9.361233e+03
1753	0.004247	277.263000	6.983115e+04	9.361236e+03
1754	0.005662	277.303986	8.987847e+04	9.364905e+03
1755	0.007077	277.351013	1.099258e+05	9.368574e+03
1756	0.008491	277.389008	1.306347e+05	1.001392e+04
1757	0.009905	277.441986	1.506820e+05	1.001758e+04
1758	0.011319	277.489014	1.707293e+05	1.002125e+04
1759	0.012732	277.540009	1.907766e+05	1.002492e+04
1760	0.014146	277.605011	2.108201e+05	1.002492e+04

In [129]:
```
df_hist_co2_ppm_mod.to_csv(
    "Data/Export/df_hist_co2_ppm_mod.csv",
    encoding = 'ANSI')
```

In [130]:
```
df_hist_ch4_ppb_mod = pd.concat([df_hist_rf.iloc[1:,2],
                                 df_hist_ghg_2014.iloc[1:,1],
                                 df_hist_em.loc[1751:,[
                                     'Earth CH4 (tot) (Gg/yr)',
                                     'Earth CH4 (fuel) (Gg/yr)']]], axis=1)

df_hist_ch4_ppb_mod.iloc[-7:-2,1] = df_hist_ghg_2019.iloc[-5:,1]


df_hist_ch4_ppb_mod
```

Out[130]:

Year	Hist CH4 (W/m²)	CH4 (ppb)	Earth CH4 (tot) (Gg/yr)	Earth CH4 (fuel) (Gg/yr)
1751	0.000508	731.838196	27003.076	5412.5472
1752	0.001016	732.899963	27076.927	5426.6235
1753	0.001524	733.634033	27150.708	5440.6497
1754	0.002031	734.202026	27224.467	5454.6348
1755	0.002538	734.736084	27298.180	5468.5741
1756	0.003045	735.421021	27376.032	5483.2574
1757	0.003552	735.893005	27449.654	5497.1054
1758	0.004058	736.206970	27523.230	5510.9076
1759	0.004564	736.098877	27596.760	5524.6642
1760	0.005069	735.484192	27670.221	5538.3706

In [131]:
```
df_hist_ch4_ppb_mod.to_csv(
    "Data/Export/df_hist_ch4_ppb_mod.csv",
    encoding = 'ANSI')
```

In [132]:
```
df_proj_rf_119.to_csv(
    "Data/Export/df_proj_rf_119.csv",
    encoding = 'ANSI')
```

In [133]:
```
df_proj_rf_126.to_csv(
    "Data/Export/df_proj_rf_126.csv",
    encoding = 'ANSI')
```

In [134]:
```
df_proj_rf_245.to_csv(
    "Data/Export/df_proj_rf_245.csv",
    encoding = 'ANSI')
```

In [135]:
```
df_proj_rf_370.to_csv(
    "Data/Export/df_proj_rf_370.csv",
    encoding = 'ANSI')
```

In [136]:
```
df_proj_rf_585.to_csv(
    "Data/Export/df_proj_rf_585.csv",
    encoding = 'ANSI')
```

In [137]:
```
df_hist_ch4_ppb_mod.to_csv(
    "Data/Export/df_hist_ch4_ppb_mod.csv",
    encoding = 'ANSI')
```

In [138]:
```
df_proj_ghg_119.to_csv(
    "Data/Export/df_proj_ghg_119.csv",
    encoding = 'ANSI')
```

In [139]:
```
df_proj_ghg_126.to_csv(
    "Data/Export/df_proj_ghg_126.csv",
    encoding = 'ANSI')
```

In [140]:
```
df_proj_ghg_245.to_csv(
    "Data/Export/df_proj_ghg_245.csv",
    encoding = 'ANSI')
```

In [141]:
```
df_proj_ghg_370.to_csv(
    "Data/Export/df_proj_ghg_370.csv",
    encoding = 'ANSI')
```

In [142]:
```
df_proj_ghg_585.to_csv(
    "Data/Export/df_proj_ghg_585.csv",
    encoding = 'ANSI')
```

In [143]:
```
df_hist_em_reg.to_csv(
    "Data/Export/df_hist_em_reg.csv",
    encoding = 'ANSI')
```

In [144]:
```
df_proj_em_co2.to_csv(
    "Data/Export/df_proj_em_co2.csv",
    encoding = 'ANSI')
```

In [145]:
```
df_proj_em_ch4.to_csv(
    "Data/Export/df_proj_em_ch4.csv",
    encoding = 'ANSI')
```

In [146]:
```
df_hist_elec_yr.to_csv(
    "Data/Export/df_hist_elec_yr.csv",
    encoding = 'ANSI')
```

In [147]:
```
df_hist_elec_yr_reg.to_csv(
    "Data/Export/df_hist_elec_yr_reg.csv",
    encoding = 'ANSI')
```

In [148]:
```
df_hist_dem_yr_reg.to_csv(
    "Data/Export/df_hist_dem_yr_reg.csv",
    encoding = 'ANSI')
```

In [149]:
```
df_proj_dem_stat_yr.to_csv(
    "Data/Export/df_proj_dem_stat_yr.csv",
    encoding = 'ANSI')
```

In [150]:
```
df_proj_dem_ann_yr.to_csv(
    "Data/Export/df_proj_dem_ann_yr.csv",
    encoding = 'ANSI')
```

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