Electricity related CO2 emission

Author: Apoorva Arbooj

Data Source: The EIA website is used as data source for this notebook.

Data Link: [EIA electricity data (https://www.eia.gov/electricity/data/eia923/)](https://www.eia.gov/electricity/data/eia923/)

Data description:
The survey Form EIA-923 collects detailed electric power data -- monthly and annually -- on electricity generation, fuel consumption, fossil fuel stocks, and receipts at the power plant and prime mover level. Specific survey information provided:

- Schedule 2 - fuel receipts and costs
- Schedules 3A & 5A - generator data including generation, fuel consumption and stocks
- Schedule 4 - fossil fuel stocks
- Schedules 6 & 7 - non-utility source and disposition of electricity
- Schedules 8A-F - environmental data

Import Libraries and mount the drive

In [1]:

```
import urllib.request
from bs4 import BeautifulSoup
from io import BytesIO
from urllib.request import urlopen
from zipfile import ZipFile
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import plotly
```

In [2]:

```
# Ensuring charts appear when converting to HTML
plotly.offline.init_notebook_mode(connected=True)
```

In [3]:
```
# Mount the drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=Tru
e).
```

Data Gathering

YEAR WISE ELECTRICITY DATA

Methodology :The data is extracted from the website by web scraping The webpage has links which enables downloading data in a compressed (zip) format. The links are arranged according to year.

In [4]:
```
# Save the web page url in the 'data_url' variable
data_url = "https://www.eia.gov/electricity/data/eia923/"
```

In [5]:
```
# Open the webpage using the url library and save the html in the 'webpage' variable
webpage = urllib.request.urlopen(data_url)
```

In [6]:
```
# Parse the html in the 'webpage' variable, and store it in Beautiful Soup format
soup = BeautifulSoup(webpage)
```

In [7]:
```
# Print the title of the web page
print(soup.title)
```

```
<title>Form EIA-923 detailed data with previous form data (EIA-906/920)</title>
```

In [8]:
```
# Get all the links on the webpage
# A link in html is contained in the anchor tag (<a>)
all_links = soup.find_all('a')
```

In [9]:
```
# Create a dictionary to store the download links on webpage
# The dictionary, data_link will be of the form
# data_links = {year:year_link}
data_links = {}
for link in all_links:
    # The required links have the year as their title
    if link.get('title') and link.get('title').strip() in map(str,range(2001,2020)):
        data_links[link.get('title').strip()] = link.get('href')
```

In [10]:
```
# Checking if the data_links dictionary has been created successfully
data_links
```

Out[10]:
```
{'2001': 'archive/xls/f906920_2001.zip',
 '2002': 'archive/xls/f906920_2002.zip',
 '2003': 'archive/xls/f906920_2003.zip',
 '2004': 'archive/xls/f906920_2004.zip',
 '2005': 'archive/xls/f906920_2005.zip',
 '2006': 'archive/xls/f906920_2006.zip',
 '2007': 'archive/xls/f906920_2007.zip',
 '2008': 'archive/xls/f923_2008.zip',
 '2009': 'archive/xls/f923_2009.zip',
 '2010': 'archive/xls/f923_2010.zip',
 '2011': 'archive/xls/f923_2011.zip',
 '2012': 'archive/xls/f923_2012.zip',
 '2013': 'archive/xls/f923_2013.zip',
 '2014': 'archive/xls/f923_2014.zip',
 '2015': 'archive/xls/f923_2015.zip',
 '2016': 'archive/xls/f923_2016.zip',
 '2017': 'archive/xls/f923_2017.zip',
 '2018': 'archive/xls/f923_2018.zip',
 '2019': 'archive/xls/f923_2019.zip'}
```

In [11]:
```
# Update the data_links dictionary with the complete downloadable url link
for k,v in data_links.items():
  data_links[k] = "https://www.eia.gov/electricity/data/eia923/"+v
```

In [12]:
```python
# Check the dictionary
data_links
```

Out[12]:
```
{'2001': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f906920_2001.zip',
 '2002': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f906920_2002.zip',
 '2003': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f906920_2003.zip',
 '2004': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f906920_2004.zip',
 '2005': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f906920_2005.zip',
 '2006': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f906920_2006.zip',
 '2007': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f906920_2007.zip',
 '2008': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f923_2008.zip',
 '2009': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f923_2009.zip',
 '2010': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f923_2010.zip',
 '2011': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f923_2011.zip',
 '2012': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f923_2012.zip',
 '2013': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f923_2013.zip',
 '2014': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f923_2014.zip',
 '2015': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f923_2015.zip',
 '2016': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f923_2016.zip',
 '2017': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f923_2017.zip',
 '2018': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f923_2018.zip',
 '2019': 'https://www.eia.gov/electricity/data/eia923/archive/xls/f923_2019.zip'}
```

In [13]:
```
# Unzipping without saving the zip file
# This cell of code will create a folder for each year and extract the files respectively
for k,v in data_links.items():
  print(f'Now downloading data for: {k}')
  zipurl = v
  with urlopen(zipurl) as zipresp:
    with ZipFile(BytesIO(zipresp.read())) as zfile:
      zfile.extractall('/content/drive/MyDrive/Colab Notebooks/ElectricityCO2Emission/AllData/'+k)
```

```
Now downloading data for: 2019
Now downloading data for: 2018
Now downloading data for: 2017
Now downloading data for: 2016
Now downloading data for: 2015
Now downloading data for: 2014
Now downloading data for: 2013
Now downloading data for: 2012
Now downloading data for: 2011
Now downloading data for: 2010
Now downloading data for: 2009
Now downloading data for: 2008
Now downloading data for: 2007
Now downloading data for: 2006
Now downloading data for: 2005
Now downloading data for: 2004
Now downloading data for: 2003
Now downloading data for: 2002
Now downloading data for: 2001
```

CO2 emission data for fuels from EIA website

Perform web scrapping to get the CO2 emitted in pounds per MMBtu of fuel

Link: [CO2 emissions (https://www.eia.gov/environment/emissions/co2_vol_mass.php)](https://www.eia.gov/environment/emissions/co2_vol_mass.php)

In [14]:
```
fuel_soup = BeautifulSoup(urllib.request.urlopen("https://www.eia.gov/environment/emissions/co2_vol_mass.php"))
```

In [15]:
```
fuel_CO2_emission = dict()
for tr in fuel_soup.find_all('tr')[2:]:
    tds = tr.find_all('td')
    if(len(tds)==5):
      fuel_CO2_emission[" ".join(tds[0].text.lower().split())]=float(tds[3].text)
```

In [16]:
```
print(f'The no. of fuels for which we have the CO2 emissions in pounds per MMBtu is: {len(fuel_CO2_emission.keys())}')
```

```
The no. of fuels for which we have the CO2 emissions in pounds per MMBtu is: 21
```

In [17]:
```
fuel_CO2_emission
```

Out[17]:
```
{'anthracite': 228.6,
 'asphalt and road oil': 166.12,
 'aviation gas': 152.46,
 'bituminous': 205.4,
 'coal (all types)': 211.06,
 'coke': 250.59,
 'diesel and home heating fuel (distillate fuel oil)': 163.45,
 'gasoline': 155.77,
 'jet fuel': 159.25,
 'kerosene': 161.35,
 'lignite': 216.24,
 'lubricants': 163.29,
 'naphthas for petrochemical feedstock use': 149.95,
 'natural gas': 116.65,
 'other oils for petrochemical feedstock use': 163.05,
 'petroleum coke': 225.13,
 'propane': 138.63,
 'residual heating fuel (businesses only)': 165.55,
 'special naphthas (solvents)': 159.57,
 'subbituminous': 214.13,
 'waxes': 160.06}
```

Get fuel description from sheet and create the final dataframe to refer for CO2 emissions

- **Year** : 2019
- **File Name**: EIA923_Schedules_2_3_4_5_M_12_2019_Final_Revision.xlsx
- **Sheet name in file**: Page 7 File Layout
- **Description**: This sheet has description for the fuel codes used in the dataset.

In [18]:
```
sheet_fuel_data_2019 = pd.read_excel("/content/drive/MyDrive/Colab Notebooks/ElectricityCO2Emission/AllData/2019/EIA923_Schedul
es_2_3_4_5_M_12_2019_Final_Revision.xlsx", sheet_name="Page 7 File Layout",skiprows=67, skipfooter=(767-108))
```

In [19]:
```
# Creating a copy of the sheet_fuel_data_2019 so that changes and be reverted by using the 'sheet_fuel_data_2019' dataframe aga
in
# Reducing frequent /re-loading of data from file
CO2_fuel_data_2019 = sheet_fuel_data_2019.copy()
```

In [20]:
```
# Checking the data in 'CO2_fuel_data_2019'
CO2_fuel_data_2019.head()
```

Out[20]:

	Reported Fuel Type Code	The fuel code reported to EIA.Two or three letter alphanumeric:
0	AB	Agricultural By-Products
1	ANT	Anthracite Coal
2	BFG	Blast Furnace Gas
3	BIT	Bituminous Coal
4	BLQ	Black Liquor

In [21]:
```
# Rename the column with a shorter name
CO2_fuel_data_2019.rename(columns={"The fuel code reported to EIA.Two or three letter alphanumeric:":"description"},inplace=Tru
e)

# Convert the descriptions to lowercase
CO2_fuel_data_2019["description"] = CO2_fuel_data_2019["description"].str.lower()

# Rename the 'Reported Fuel Type Code' column so that it is in sync with the column name in sheet1
CO2_fuel_data_2019.rename(columns={"Reported Fuel Type Code":"Reported_Fuel_Type_Code"},inplace=True)

# Remove any trailing or leading whitespaces
CO2_fuel_data_2019["Reported_Fuel_Type_Code"] = CO2_fuel_data_2019["Reported_Fuel_Type_Code"].str.strip()

# Check if the above changes are replicated
CO2_fuel_data_2019.head()
```

Out[21]:

	Reported_Fuel_Type_Code	description
0	AB	agricultural by-products
1	ANT	anthracite coal
2	BFG	blast furnace gas
3	BIT	bituminous coal
4	BLQ	black liquor

In [22]:
```
# There are some fuels in the dataset for which the CO2 emission values are not present of the EIA website
# Initialise the 'CO2_emission_pound_per_MMBtu' column with 1
# Assumption: (defualt) 1 pound of CO2 is emitted per million Btu of fuel
CO2_fuel_data_2019["CO2_emission_pound_per_MMBtu"] = np.float(1)
```

In [23]:
```
# Create a column named 'mapped_fuel_name' to verify if the CO2 emission value was correctly referenced
# Initialize the column with empty string
CO2_fuel_data_2019["mapped_fuel_name"] = ''

# for each (k: fuel and v: CO2 emission value) pair in the dictionary 'fuel_CO2_emission'
for k,v in fuel_CO2_emission.items():
    # if CO2_fuel_data_2019["description"] / fuel description column in 'CO2_fuel_data_2019' dataframe contains the k (fuel) fr
om the dictionary 'fuel_CO2_emission'
    # then assign / fill the column 'CO2_emission_pound_per_MMBtu' in 'CO2_fuel_data_2019' dataframe with v (CO2 emission valu
e) from the dictionary 'fuel_CO2_emission'
    CO2_fuel_data_2019.loc[CO2_fuel_data_2019["description"].str.contains(k), "CO2_emission_pound_per_MMBtu"] = v

    CO2_fuel_data_2019.loc[CO2_fuel_data_2019["description"].str.contains(k), "mapped_fuel_name"] = k
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/strings.py:2001: UserWarning:

This pattern has match groups. To actually get the groups, use str.extract.
```

In [24]:
```
# Check how the above code cell worked
CO2_fuel_data_2019
```

Out[24]:

	Reported_Fuel_Type_Code	description	CO2_emission_pound_per_MMBtu	mapped_fuel_name
0	AB	agricultural by-products	1.00	
1	ANT	anthracite coal	228.60	anthracite
2	BFG	blast furnace gas	1.00	
3	BIT	bituminous coal	205.40	bituminous
4	BLQ	black liquor	1.00	
5	DFO	distillate fuel oil. including diesel, no. 1, ...	1.00	
6	GEO	geothermal	1.00	
7	JF	jet fuel	159.25	jet fuel
8	KER	kerosene	161.35	kerosene
9	LFG	landfill gas	1.00	
10	LIG	lignite coal	216.24	lignite
11	MSB	biogenic municipal solid waste	1.00	
12	MSN	non-biogenic municipal solid waste	1.00	
13	MWH	electricity used for energy storage	1.00	
14	NG	natural gas	116.65	natural gas
15	NUC	nuclear. including uranium, plutonium, and tho...	1.00	
16	OBG	other biomass gas. including digester gas, met...	1.00	
17	OBL	other biomass liquids	1.00	
18	OBS	other biomass solids	1.00	
19	OG	other gas	1.00	
20	OTH	other fuel	1.00	
21	PC	petroleum coke	250.59	coke
22	PG	gaseous propane	138.63	propane
23	PUR	purchased steam	1.00	
24	RC	refined coal	1.00	
25	RFO	residual fuel oil. including no. 5 & 6 fuel oi...	1.00	
26	SC	coal-based synfuel. including briquettes, pell...	1.00	
27	SGC	coal-derived synthesis gas	1.00	

	Reported_Fuel_Type_Code	description	CO2_emission_pound_per_MMBtu	mapped_fuel_name
28	SGP	synthesis gas from petroleum coke	250.59	coke
29	SLW	sludge waste	1.00	
30	SUB	subbituminous coal	214.13	subbituminous
31	SUN	solar	1.00	
32	TDF	tire-derived fuels	1.00	
33	WAT	water at a conventional hydroelectric turbine ...	1.00	
34	WC	waste/other coal. including anthracite culm, b...	216.24	lignite
35	WDL	wood waste liquids, excluding black liquor. in...	1.00	
36	WDS	wood/wood waste solids. including paper pellet...	1.00	
37	WH	waste heat not directly attributed to a fuel s...	1.00	
38	WND	wind	1.00	
39	WO	waste/other oil. including crude oil, liquid b...	138.63	propane

The pair ('*diesel and home heating fuel (distillate fuel oil)': 163.45*) in the fuel_CO2_emission dictionary, did not find the exact match in the CO2_fuel_data_2019["description"] column.
Because CO2_fuel_data_2019["description"] column has *Distillate Fuel Oil. Including diesel, No. 1, No. 2, and No. 4 fuel oils*.

In [25]:
```
CO2_fuel_data_2019.loc[CO2_fuel_data_2019["description"].str.contains('diesel'), "CO2_emission_pound_per_MMBtu"] = 163.45
CO2_fuel_data_2019.loc[CO2_fuel_data_2019["description"].str.contains('diesel'), "mapped_fuel_name"] = "diesel"
```

In [26]:

```
CO2_fuel_data_2019
```

Out[26]:

	Reported_Fuel_Type_Code	description	CO2_emission_pound_per_MMBtu	mapped_fuel_name
0	AB	agricultural by-products	1.00	
1	ANT	anthracite coal	228.60	anthracite
2	BFG	blast furnace gas	1.00	
3	BIT	bituminous coal	205.40	bituminous
4	BLQ	black liquor	1.00	
5	DFO	distillate fuel oil. including diesel, no. 1, ...	163.45	diesel
6	GEO	geothermal	1.00	
7	JF	jet fuel	159.25	jet fuel
8	KER	kerosene	161.35	kerosene
9	LFG	landfill gas	1.00	
10	LIG	lignite coal	216.24	lignite
11	MSB	biogenic municipal solid waste	1.00	
12	MSN	non-biogenic municipal solid waste	1.00	
13	MWH	electricity used for energy storage	1.00	
14	NG	natural gas	116.65	natural gas
15	NUC	nuclear. including uranium, plutonium, and tho...	1.00	
16	OBG	other biomass gas. including digester gas, met...	1.00	
17	OBL	other biomass liquids	1.00	
18	OBS	other biomass solids	1.00	
19	OG	other gas	1.00	
20	OTH	other fuel	1.00	
21	PC	petroleum coke	250.59	coke
22	PG	gaseous propane	138.63	propane
23	PUR	purchased steam	1.00	
24	RC	refined coal	1.00	
25	RFO	residual fuel oil. including no. 5 & 6 fuel oi...	1.00	
26	SC	coal-based synfuel. including briquettes, pell...	1.00	
27	SGC	coal-derived synthesis gas	1.00	

	Reported_Fuel_Type_Code	description	CO2_emission_pound_per_MMBtu	mapped_fuel_name
28	SGP	synthesis gas from petroleum coke	250.59	coke
29	SLW	sludge waste	1.00	
30	SUB	subbituminous coal	214.13	subbituminous
31	SUN	solar	1.00	
32	TDF	tire-derived fuels	1.00	
33	WAT	water at a conventional hydroelectric turbine ...	1.00	
34	WC	waste/other coal. including anthracite culm, b...	216.24	lignite
35	WDL	wood waste liquids, excluding black liquor. in...	1.00	
36	WDS	wood/wood waste solids. including paper pellet...	1.00	
37	WH	waste heat not directly attributed to a fuel s...	1.00	
38	WND	wind	1.00	
39	WO	waste/other oil. including crude oil, liquid b...	138.63	propane

In [27]:
```
CO2_fuel_data_2019.to_excel("fuel_CO2_emission.xlsx")
```

2019 Data

- **Year** : 2019
- **File Name**: EIA923_Schedules_2_3_4_5_M_12_2019_Final_Revision.xlsx
- **Sheet name in file**: Page 1 Generation and Fuel Data
- **Description**: This sheet has electricity generation and fuel data.

In [28]:
```
data_2019 = pd.read_excel("/content/drive/MyDrive/Colab Notebooks/ElectricityCO2Emission/AllData/2019/EIA923_Schedules_2_3_4_5_M_12_2019_Final_Revision.xlsx", sheet_name="Page 1 Generation and Fuel Data",skiprows=5)
```

In [29]:
```
print(f'In the 2019 dataset\n# rows: {data_2019.shape[0]}\n# columns: {data_2019.shape[1]}')
```

```
In the 2019 dataset
# rows: 14517
# columns: 97
```

In [30]:
```
# Checking the datatypes of columns in the dataset
data_2019.dtypes
```

Out[30]:
```
Plant Id                                    int64
Combined Heat And\nPower Plant             object
Nuclear Unit Id                            object
Plant Name                                 object
Operator Name                              object
                                            ...
Electric Fuel Consumption\nQuantity         int64
Total Fuel Consumption\nMMBtu               int64
Elec Fuel Consumption\nMMBtu                int64
Net Generation\n(Megawatthours)           float64
YEAR                                        int64
Length: 97, dtype: object
```

In [31]:
```
# Checking the dataset
data_2019.head()
```

Out[31]:

	Plant Id	Combined Heat And\nPower Plant	Nuclear Unit Id	Plant Name	Operator Name	Operator Id	Plant State	Census Region	NERC Region	Reserved	NAICS Code	EIA Sector Number	Sector Name	Reported\nPrime Mover	Reported\nFue Type Code
0	1	N	.	Sand Point	TDX Sand Point Generating, LLC	63560	AK	PACN	NaN	NaN	22	2	NAICS-22 Non-Cogen	IC	DFC
1	1	N	.	Sand Point	TDX Sand Point Generating, LLC	63560	AK	PACN	NaN	NaN	22	2	NAICS-22 Non-Cogen	WT	WNE
2	2	N	.	Bankhead Dam	Alabama Power Co	195	AL	ESC	SERC	NaN	22	1	Electric Utility	HY	WAT
3	3	N	.	Barry	Alabama Power Co	195	AL	ESC	SERC	NaN	22	1	Electric Utility	CA	NG
4	3	N	.	Barry	Alabama Power Co	195	AL	ESC	SERC	NaN	22	1	Electric Utility	CT	NG

5 rows × 97 columns

In [32]:
```
# Extract columns that are required for further calculations
data_2019_workspace_dataframe = pd.concat([data_2019.iloc[:,[0,3,4,6,12,14,15,18]].copy(),data_2019.iloc[:,67:91].copy(),data_2
019.iloc[:,94:].copy()],axis=1)
```

In [33]:
```
# Checking if the required columns are extracted in the dataframe
data_2019_workspace_dataframe.head()
```

Out[33]:

	Plant Id	Plant Name	Operator Name	Plant State	Sector Name	Reported\nFuel Type Code	AER\nFuel Type Code	Physical\nUnit Label	Elec_MMBtu\nJanuary	Elec_MMBtu\nFebruary	Elec_MMBtu\nMarch
0	1	Sand Point	TDX Sand Point Generating, LLC	AK	NAICS-22 Non-Cogen	DFO	DFO	barrels	2045	2283	2260
1	1	Sand Point	TDX Sand Point Generating, LLC	AK	NAICS-22 Non-Cogen	WND	WND	NaN	784	717	804
2	2	Bankhead Dam	Alabama Power Co	AL	Electric Utility	WAT	HYC	NaN	0	0	0
3	3	Barry	Alabama Power Co	AL	Electric Utility	NG	NG	mcf	26661	49009	63913
4	3	Barry	Alabama Power Co	AL	Electric Utility	NG	NG	mcf	5040187	4735032	5133988

In [34]:
```
# Check column names
data_2019_workspace_dataframe.columns
```

Out[34]:
```
Index(['Plant Id', 'Plant Name', 'Operator Name', 'Plant State', 'Sector Name',
       'Reported\nFuel Type Code', 'AER\nFuel Type Code',
       'Physical\nUnit Label', 'Elec_MMBtu\nJanuary', 'Elec_MMBtu\nFebruary',
       'Elec_MMBtu\nMarch', 'Elec_MMBtu\nApril', 'Elec_MMBtu\nMay',
       'Elec_MMBtu\nJune', 'Elec_MMBtu\nJuly', 'Elec_MMBtu\nAugust',
       'Elec_MMBtu\nSeptember', 'Elec_MMBtu\nOctober', 'Elec_MMBtu\nNovember',
       'Elec_MMBtu\nDecember', 'Netgen\nJanuary', 'Netgen\nFebruary',
       'Netgen\nMarch', 'Netgen\nApril', 'Netgen\nMay', 'Netgen\nJune',
       'Netgen\nJuly', 'Netgen\nAugust', 'Netgen\nSeptember',
       'Netgen\nOctober', 'Netgen\nNovember', 'Netgen\nDecember',
       'Elec Fuel Consumption\nMMBtu', 'Net Generation\n(Megawatthours)',
       'YEAR'],
      dtype='object')
```

In [35]:
```
# Replace '\n' character and space in the column name with '_'
for c in data_2019_workspace_dataframe.columns:
    new_column_name = (c.replace("\n","_")).replace(" ","_")
    data_2019_workspace_dataframe.rename(columns={c:new_column_name},inplace=True)
```

In [36]:
```
# Check if the column names were replaces and renamed correctly
data_2019_workspace_dataframe.columns
```

Out[36]:
```
Index(['Plant_Id', 'Plant_Name', 'Operator_Name', 'Plant_State', 'Sector_Name',
       'Reported_Fuel_Type_Code', 'AER_Fuel_Type_Code', 'Physical_Unit_Label',
       'Elec_MMBtu_January', 'Elec_MMBtu_February', 'Elec_MMBtu_March',
       'Elec_MMBtu_April', 'Elec_MMBtu_May', 'Elec_MMBtu_June',
       'Elec_MMBtu_July', 'Elec_MMBtu_August', 'Elec_MMBtu_September',
       'Elec_MMBtu_October', 'Elec_MMBtu_November', 'Elec_MMBtu_December',
       'Netgen_January', 'Netgen_February', 'Netgen_March', 'Netgen_April',
       'Netgen_May', 'Netgen_June', 'Netgen_July', 'Netgen_August',
       'Netgen_September', 'Netgen_October', 'Netgen_November',
       'Netgen_December', 'Elec_Fuel_Consumption_MMBtu',
       'Net_Generation_(Megawatthours)', 'YEAR'],
      dtype='object')
```

In [37]:
```
# Monthly data has '.' so replace that with zero
# Replacing with zero won't impact the further calculations
for i in range(8,32):
    data_2019_workspace_dataframe.iloc[:,i].replace(to_replace={'.':0}, inplace=True)
```

In [38]:
```
# Checking the datatypes of all columns
data_2019_workspace_dataframe.dtypes
```

Out[38]:
```
Plant_Id                             int64
Plant_Name                          object
Operator_Name                       object
Plant_State                         object
Sector_Name                         object
Reported_Fuel_Type_Code             object
AER_Fuel_Type_Code                  object
Physical_Unit_Label                 object
Elec_MMBtu_January                   int64
Elec_MMBtu_February                  int64
Elec_MMBtu_March                     int64
Elec_MMBtu_April                     int64
Elec_MMBtu_May                       int64
Elec_MMBtu_June                      int64
Elec_MMBtu_July                      int64
Elec_MMBtu_August                    int64
Elec_MMBtu_September                 int64
Elec_MMBtu_October                   int64
Elec_MMBtu_November                  int64
Elec_MMBtu_December                  int64
Netgen_January                     float64
Netgen_February                    float64
Netgen_March                       float64
Netgen_April                       float64
Netgen_May                         float64
Netgen_June                        float64
Netgen_July                        float64
Netgen_August                      float64
Netgen_September                   float64
Netgen_October                     float64
Netgen_November                    float64
Netgen_December                    float64
Elec_Fuel_Consumption_MMBtu          int64
Net_Generation_(Megawatthours)     float64
YEAR                                 int64
dtype: object
```

Fuels used in 2019

In [39]:
```
list_fuel_type_code = list(data_2019_workspace_dataframe["Reported_Fuel_Type_Code"].unique())
print(f'The dataset has {len(list_fuel_type_code)} unique types of fuel.\nThe list of fuel type code is:\n{list_fuel_type_cod
e}')
```

```
The dataset has 39 unique types of fuel.
The list of fuel type code is:
['DFO', 'WND', 'WAT', 'NG', 'BIT', 'SUB', 'NUC', 'LIG', 'PG', 'RC', 'AB', 'WDS', 'RFO', 'LFG', 'PC', 'SUN', 'OBG', 'GEO', 'MW
H', 'OG', 'WO', 'JF', 'KER', 'OTH', 'WC', 'SGC', 'OBS', 'TDF', 'BFG', 'MSB', 'MSN', 'SC', 'BLQ', 'WH', 'OBL', 'SLW', 'PUR', 'WD
L', 'SGP']
```

In [40]:
```
# Count the number of times a fuel appears in the dataset and store it in a dictionary
fuel_frequency = dict(data_2019_workspace_dataframe["Reported_Fuel_Type_Code"].value_counts())
```

In [41]:
```
# Convert the fuel_frequency dictionary to a dataframe
df_fuel_frequency = pd.DataFrame({"Reported_Fuel_Type_Code":fuel_frequency.keys(),"Frequency":fuel_frequency.values()})
```

In [42]:
```
# To find which fuel type is absent in the dataset
for f in CO2_fuel_data_2019["Reported_Fuel_Type_Code"].unique():
  if f not in list_fuel_type_code:
    print(f'{f} is absent from the "Electricity Generation and Fuel" dataset.')
# ANT is anthracite coal
```

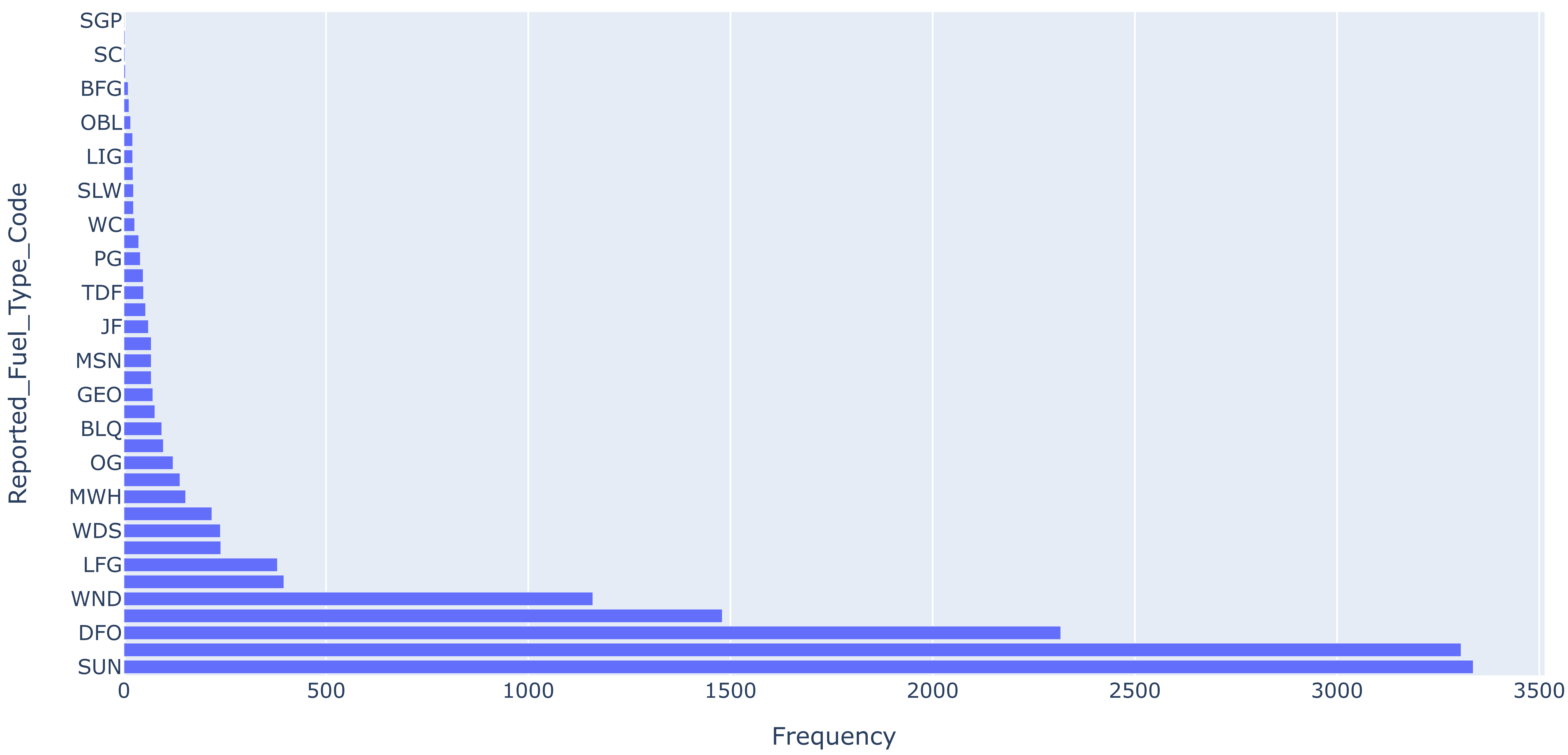
```
ANT is absent from the "Electricity Generation and Fuel" dataset.
```

In [43]:
```
df_fuel_frequency.head()
```

Out[43]:

	Reported_Fuel_Type_Code	Frequency
0	SUN	3337
1	NG	3307
2	DFO	2317
3	WAT	1480
4	WND	1160

In [44]:
```
# Plot the 'df_fuel_frequency' dataframe
# Check fuels used for electricity generation from least to most
fig = px.bar(df_fuel_frequency, x="Frequency", y="Reported_Fuel_Type_Code", orientation='h')
fig.show()
```



Note: Solar energy topped the electricity generation in 2019 followed by 'Natural Gas' and 'DFO (Distillate Fuel Oil. Including diesel, No. 1, No. 2, and No. 4 fuel oils.)'

Find CO2 emission in Metric tons and Carbon Intensity in MtCO2/MWh for 2019 dataset

In [45]:
```python
# Display the required 2019 data
data_2019_workspace_dataframe
```

Out[45]:

	Plant_Id	Plant_Name	Operator_Name	Plant_State	Sector_Name	Reported_Fuel_Type_Code	AER_Fuel_Type_Code	Physical_Unit_Label	Elec_MMBtu_Ja
0	1	Sand Point	TDX Sand Point Generating, LLC	AK	NAICS-22 Non-Cogen	DFO	DFO	barrels	
1	1	Sand Point	TDX Sand Point Generating, LLC	AK	NAICS-22 Non-Cogen	WND	WND	NaN	
2	2	Bankhead Dam	Alabama Power Co	AL	Electric Utility	WAT	HYC	NaN	
3	3	Barry	Alabama Power Co	AL	Electric Utility	NG	NG	mcf	
4	3	Barry	Alabama Power Co	AL	Electric Utility	NG	NG	mcf	5(
...	...	...	...	...	...	...	...	...	
14512	99999	State-Fuel Level Increment	State-Fuel Level Increment	SD	Industrial NAICS Cogen	WDS	WWW	short tons	
14513	99999	State-Fuel Level Increment	State-Fuel Level Increment	SC	Electric Utility	WO	WOO	barrels	
14514	99999	State-Fuel Level Increment	State-Fuel Level Increment	CA	NAICS-22 Non-Cogen	WND	WND	NaN	
14515	99999	State-Fuel Level Increment	State-Fuel Level Increment	MI	NAICS-22 Non-Cogen	WND	WND	NaN	
14516	99999	State-Fuel Level Increment	State-Fuel Level Increment	TX	NAICS-22 Non-Cogen	WND	WND	NaN	

14517 rows × 35 columns

In [46]:
```python
# Merge the '2019 electricity generation and fuel' data with 'CO2 emission for fuels' data
data_2019_workspace_dataframe = pd.merge(data_2019_workspace_dataframe,CO2_fuel_data_2019,how='left',on=["Reported_Fuel_Type_Co
de"])
```

In [47]:
```
# Check the merge
data_2019_workspace_dataframe
```

Out[47]:

	Plant_Id	Plant_Name	Operator_Name	Plant_State	Sector_Name	Reported_Fuel_Type_Code	AER_Fuel_Type_Code	Physical_Unit_Label	Elec_MMBtu_Ja
0	1	Sand Point	TDX Sand Point Generating, LLC	AK	NAICS-22 Non-Cogen	DFO	DFO	barrels	
1	1	Sand Point	TDX Sand Point Generating, LLC	AK	NAICS-22 Non-Cogen	WND	WND	NaN	
2	2	Bankhead Dam	Alabama Power Co	AL	Electric Utility	WAT	HYC	NaN	
3	3	Barry	Alabama Power Co	AL	Electric Utility	NG	NG	mcf	
4	3	Barry	Alabama Power Co	AL	Electric Utility	NG	NG	mcf	5(
...	...	...	...	...	...	...	...	...	
14512	99999	State-Fuel Level Increment	State-Fuel Level Increment	SD	Industrial NAICS Cogen	WDS	WWW	short tons	
14513	99999	State-Fuel Level Increment	State-Fuel Level Increment	SC	Electric Utility	WO	WOO	barrels	
14514	99999	State-Fuel Level Increment	State-Fuel Level Increment	CA	NAICS-22 Non-Cogen	WND	WND	NaN	
14515	99999	State-Fuel Level Increment	State-Fuel Level Increment	MI	NAICS-22 Non-Cogen	WND	WND	NaN	
14516	99999	State-Fuel Level Increment	State-Fuel Level Increment	TX	NAICS-22 Non-Cogen	WND	WND	NaN	

14517 rows × 38 columns

In [48]:
```python
# Count the null rows in the "Physical_Unit_Label" column
data_2019_workspace_dataframe["Physical_Unit_Label"].isnull().sum()
```

Out[48]:
```
6180
```

In [49]:
```python
# Change the datatype of "Physical_Unit_Label" column to string
data_2019_workspace_dataframe["Physical_Unit_Label"] = data_2019_workspace_dataframe["Physical_Unit_Label"].astype('str')
```

In [50]:
```python
# Verify the datatype
for unit in data_2019_workspace_dataframe["Physical_Unit_Label"].unique():
    print(f'{unit}\t{type(unit)}')
```

```
barrels <class 'str'>
nan     <class 'str'>
mcf     <class 'str'>
short tons      <class 'str'>
megawatthours   <class 'str'>
Mcf     <class 'str'>
```

In [51]:

```
# For rows where "Physical_Unit_Label" is null, set the "CO2_emission_pound_per_MMBtu" value to zero
# Mostly, for renewable source of energy the "Physical_Unit_Label" is null
# Also, there is no CO2 emission in case of renewable source of energy
# SO, setting the "CO2_emission_pound_per_MMBtu" value to zero
data_2019_workspace_dataframe.loc[(data_2019_workspace_dataframe["Physical_Unit_Label"] == "nan"),"CO2_emission_pound_per_MMBt
u"]=0

# Check the "CO2_emission_pound_per_MMBtu" column modification
data_2019_workspace_dataframe
```

Out[51]:

	Plant_Id	Plant_Name	Operator_Name	Plant_State	Sector_Name	Reported_Fuel_Type_Code	AER_Fuel_Type_Code	Physical_Unit_Label	Elec_MMBtu_Ja
0	1	Sand Point	TDX Sand Point Generating, LLC	AK	NAICS-22 Non-Cogen	DFO	DFO	barrels	
1	1	Sand Point	TDX Sand Point Generating, LLC	AK	NAICS-22 Non-Cogen	WND	WND	nan	
2	2	Bankhead Dam	Alabama Power Co	AL	Electric Utility	WAT	HYC	nan	
3	3	Barry	Alabama Power Co	AL	Electric Utility	NG	NG	mcf	
4	3	Barry	Alabama Power Co	AL	Electric Utility	NG	NG	mcf	5(
...	...	...	...	...	...	...	...	...	
14512	99999	State-Fuel Level Increment	State-Fuel Level Increment	SD	Industrial NAICS Cogen	WDS	WWW	short tons	
14513	99999	State-Fuel Level Increment	State-Fuel Level Increment	SC	Electric Utility	WO	WOO	barrels	
14514	99999	State-Fuel Level Increment	State-Fuel Level Increment	CA	NAICS-22 Non-Cogen	WND	WND	nan	
14515	99999	State-Fuel Level Increment	State-Fuel Level Increment	MI	NAICS-22 Non-Cogen	WND	WND	nan	
14516	99999	State-Fuel Level Increment	State-Fuel Level Increment	TX	NAICS-22 Non-Cogen	WND	WND	nan	

14517 rows × 38 columns

Based on the website information, 1 MMBtu of 'Natural Gas' emits 116.65 pounds of CO2.
Hence to find Metric tons of CO2 emitted for the Elec_Fuel_Consumption_MMBtu, we have

Annual Fuel Consumption to generate electricity in MMBtu = *data_2019_workspace_dataframe["Elec_Fuel_Consumption_MMBtu"]*
CO2 emission of the fuel in pound per MMBtu = *data_2019_workspace_dataframe["CO2_emission_pound_per_MMBtu"]*

Annual CO2 emission of the fuel used to generate electricity in pounds per MMBtu = *data_2019_workspace_dataframe["Elec_Fuel_Consumption_MMBtu"] x data_2019_workspace_dataframe["CO2_emission_pound_per_MMBtu"]*

Annual CO2 emission of the fuel used to generate electricity in Metric tons per MMBtu = *(Annual CO2 emission of the fuel used to generate electricity in pounds per MMBtu) / 2205*

In [52]:
```
data_2019_workspace_dataframe["Annual_Elect_CO2_emission_MtCO2"] = (data_2019_workspace_dataframe["Elec_Fuel_Consumption_MMBtu"
]*data_2019_workspace_dataframe["CO2_emission_pound_per_MMBtu"])/2205
```

Carbon Intensity = CO2 emitted per unit of electric energy generated
Carbon Intensity = *(Total amount of electricity related CO2 emitted in Metric tons) / (Total amount of electricity produced in Megawatthours) = Carbon Intensity in MtCO2/MWh*

In [53]:
```
# using fillna to handle 0/0 giving NaN
data_2019_workspace_dataframe["Annual_Carbon_Intensity_MtCO2_per_MWh"] = (data_2019_workspace_dataframe["Annual_Elect_CO2_emiss
ion_MtCO2"]/data_2019_workspace_dataframe["Net_Generation_(Megawatthours)"]).fillna(0)

# if above there was a (something)/0 case, then it gives inf so replace inf with 0
data_2019_workspace_dataframe.loc[data_2019_workspace_dataframe["Annual_Carbon_Intensity_MtCO2_per_MWh"] == np.inf, "Annual_Car
bon_Intensity_MtCO2_per_MWh"] = 0
data_2019_workspace_dataframe.loc[data_2019_workspace_dataframe["Annual_Carbon_Intensity_MtCO2_per_MWh"] == -np.inf, "Annual_Ca
rbon_Intensity_MtCO2_per_MWh"] = 0
```

In [54]:
```
np.where(np.isinf(data_2019_workspace_dataframe["Annual_Carbon_Intensity_MtCO2_per_MWh"]))
```

Out[54]:
```
(array([], dtype=int64),)
```

In [55]:
```
# Display the data
data_2019_workspace_dataframe.head()
```

Out[55]:

	Plant_Id	Plant_Name	Operator_Name	Plant_State	Sector_Name	Reported_Fuel_Type_Code	AER_Fuel_Type_Code	Physical_Unit_Label	Elec_MMBtu_Janua
0	1	Sand Point	TDX Sand Point Generating, LLC	AK	NAICS-22 Non-Cogen	DFO	DFO	barrels	204
1	1	Sand Point	TDX Sand Point Generating, LLC	AK	NAICS-22 Non-Cogen	WND	WND	nan	78
2	2	Bankhead Dam	Alabama Power Co	AL	Electric Utility	WAT	HYC	nan	
3	3	Barry	Alabama Power Co	AL	Electric Utility	NG	NG	mcf	2666
4	3	Barry	Alabama Power Co	AL	Electric Utility	NG	NG	mcf	504018

In [56]:
```
months = ["January","February","March","April","May","June","July","August","September","October","November","December"]
```

In [57]:
```
# Calculating monthly electricity related CO2 emission in Metric tons per MMBtu
for m in months:
    new_column = m+"_Elect_CO2_emission_MtCO2"
    existing_column = "Elec_MMBtu_"+m
    data_2019_workspace_dataframe[new_column] = (data_2019_workspace_dataframe["CO2_emission_pound_per_MMBtu"]*data_2019_worksp
ace_dataframe[existing_column])/2205
```

In [58]:
```
data_2019_workspace_dataframe.head()
```

Out[58]:

	Plant_Id	Plant_Name	Operator_Name	Plant_State	Sector_Name	Reported_Fuel_Type_Code	AER_Fuel_Type_Code	Physical_Unit_Label	Elec_MMBtu_Janua
0	1	Sand Point	TDX Sand Point Generating, LLC	AK	NAICS-22 Non-Cogen	DFO	DFO	barrels	204
1	1	Sand Point	TDX Sand Point Generating, LLC	AK	NAICS-22 Non-Cogen	WND	WND	nan	78
2	2	Bankhead Dam	Alabama Power Co	AL	Electric Utility	WAT	HYC	nan	
3	3	Barry	Alabama Power Co	AL	Electric Utility	NG	NG	mcf	2666
4	3	Barry	Alabama Power Co	AL	Electric Utility	NG	NG	mcf	504018

In [59]:
```
# Calculating monthly carbon intensity of electricity generation in Metric tons of CO2 per MegaWatthour
for m in months:
    new_column = m+"_Carbon_Intensity_MtCO2_per_MWh"
    numerator_column = m+"_Elect_CO2_emission_MtCO2"
    denominator_column = "Netgen_"+m
    data_2019_workspace_dataframe[new_column] = data_2019_workspace_dataframe[numerator_column]/data_2019_workspace_dataframe[d
enominator_column].fillna(0)
    data_2019_workspace_dataframe.loc[data_2019_workspace_dataframe[new_column] == np.inf, new_column] = 0
    data_2019_workspace_dataframe.loc[data_2019_workspace_dataframe[new_column] == -np.inf, new_column] = 0
```

In [60]:
```
data_2019_workspace_dataframe.head()
```

Out[60]:

	Plant_Id	Plant_Name	Operator_Name	Plant_State	Sector_Name	Reported_Fuel_Type_Code	AER_Fuel_Type_Code	Physical_Unit_Label	Elec_MMBtu_Januar
0	1	Sand Point	TDX Sand Point Generating, LLC	AK	NAICS-22 Non-Cogen	DFO	DFO	barrels	204
1	1	Sand Point	TDX Sand Point Generating, LLC	AK	NAICS-22 Non-Cogen	WND	WND	nan	78
2	2	Bankhead Dam	Alabama Power Co	AL	Electric Utility	WAT	HYC	nan	
3	3	Barry	Alabama Power Co	AL	Electric Utility	NG	NG	mcf	2666
4	3	Barry	Alabama Power Co	AL	Electric Utility	NG	NG	mcf	504018

In [61]:
```
data_2019_workspace_dataframe.to_csv("Final_data_2019.csv")
```

Carbon Intensity in MtCO2/MWh by Fuel type

ANNUALLY

In [62]:
```
# Get a subset of data to calculate Annual_Carbon_Intensity_MtCO2_per_MWh by fuel
df_Annual_Carbon_Intensity_by_Fuel = pd.concat([data_2019_workspace_dataframe.iloc[:,[5,39]].copy()],axis=1)
```

In [63]:
```
df_Annual_Carbon_Intensity_by_Fuel
```

Out[63]:

	Reported_Fuel_Type_Code	Annual_Carbon_Intensity_MtCO2_per_MWh
0	DFO	0.777661
1	WND	0.000000
2	WAT	-0.000000
3	NG	0.028084
4	NG	0.554370
...	...	...
14512	WDS	0.000000
14513	WO	0.661437
14514	WND	0.000000
14515	WND	0.000000
14516	WND	0.000000

14517 rows × 2 columns

In [64]:
```
# Perform summation by grouping by 'fuel code'
df_Annual_Carbon_Intensity_by_Fuel=df_Annual_Carbon_Intensity_by_Fuel.groupby(by=["Reported_Fuel_Type_Code"]).sum()
```

In [65]:
```
# reset the index
df_Annual_Carbon_Intensity_by_Fuel.reset_index(level=0,inplace=True)
```

In [66]:
```
# sort data in descending order of 'Annual_Carbon_Intensity_MtCO2_per_MWh'
df_Annual_Carbon_Intensity_by_Fuel=df_Annual_Carbon_Intensity_by_Fuel.sort_values(by=["Annual_Carbon_Intensity_MtCO2_per_MWh"],
ascending=False)
```

In [67]:
```
df_Annual_Carbon_Intensity_by_Fuel
```

Out[67]:

	Reported_Fuel_Type_Code	Annual_Carbon_Intensity_MtCO2_per_MWh
4	DFO	2289.694996
13	NG	1779.363852
2	BIT	179.726525
29	SUB	170.672445
33	WC	23.435266
7	KER	20.739059
20	PC	20.360766
6	JF	19.759274
9	LIG	17.763558
38	WO	17.271795
21	PG	10.302246
8	LFG	2.298147
35	WDS	0.893912
11	MSN	0.599355
10	MSB	0.599355
24	RFO	0.528185
15	OBG	0.437007
23	RC	0.290626
18	OG	0.244884
3	BLQ	0.240411
31	TDF	0.163967
0	AB	0.082375
16	OBL	0.050718
17	OBS	0.047392
28	SLW	0.039634
1	BFG	0.027932
34	WDL	0.007721
26	SGC	0.006983

	Reported_Fuel_Type_Code	Annual_Carbon_Intensity_MtCO2_per_MWh
22	PUR	0.000000
12	MWH	0.000000
37	WND	0.000000
36	WH	0.000000
5	GEO	0.000000
30	SUN	0.000000
32	WAT	0.000000
14	NUC	0.000000
27	SGP	0.000000
25	SC	0.000000
19	OTH	0.000000

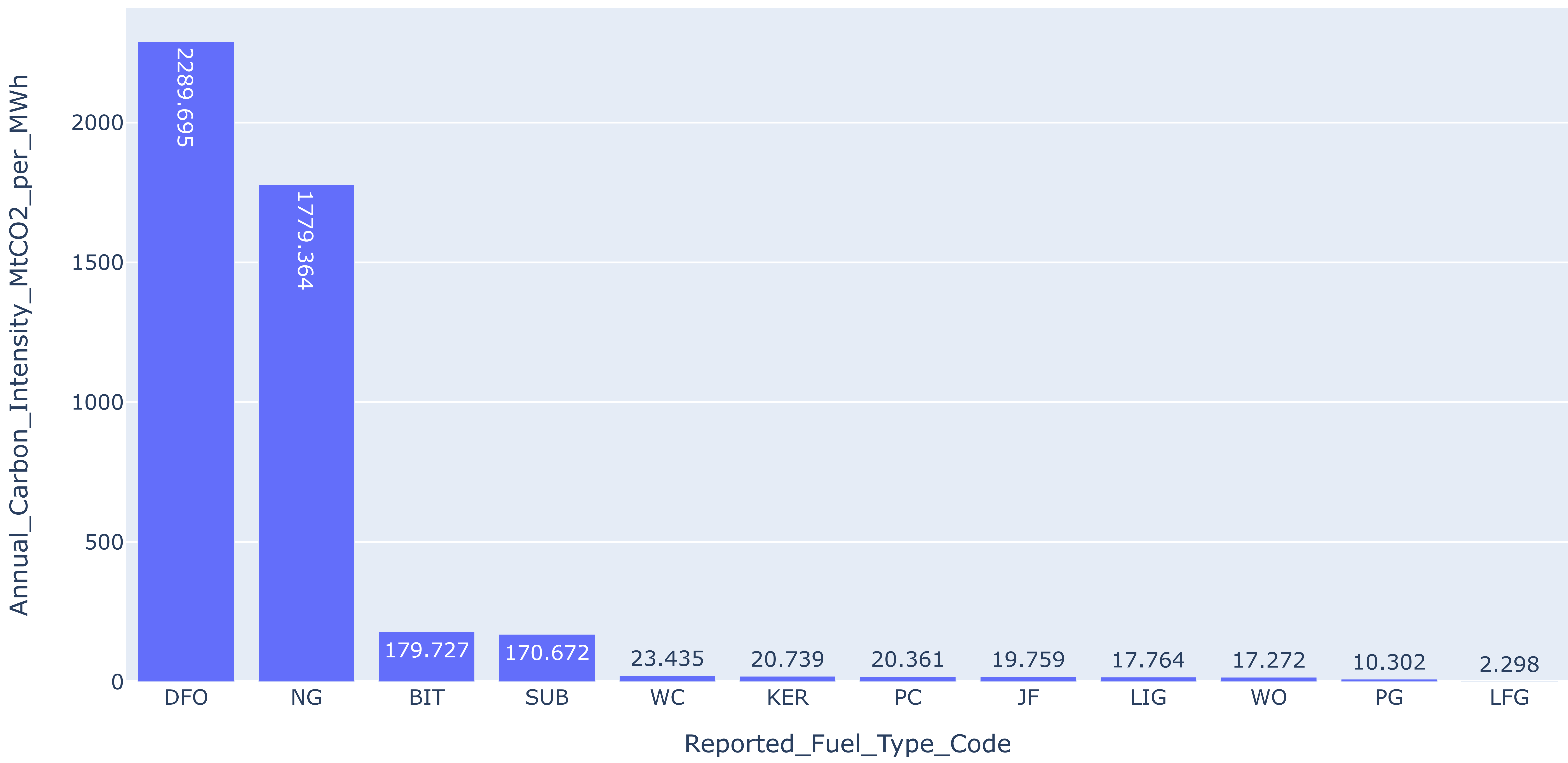
In [68]:

```
# Create a data copy where the Carbon Intensity value is greater than 1 for plotting it in a bar chart
df_Annual_Carbon_Intensity_by_Fuel_Plotting_data = df_Annual_Carbon_Intensity_by_Fuel[df_Annual_Carbon_Intensity_by_Fuel["Annua
l_Carbon_Intensity_MtCO2_per_MWh"]>1].copy()
```

In [69]:

```
# Round the float numbers to 3
df_Annual_Carbon_Intensity_by_Fuel_Plotting_data["Annual_Carbon_Intensity_MtCO2_per_MWh"] = df_Annual_Carbon_Intensity_by_Fuel_
Plotting_data["Annual_Carbon_Intensity_MtCO2_per_MWh"].round(3)
```

In [70]:
```
fig = px.bar(df_Annual_Carbon_Intensity_by_Fuel_Plotting_data,
             x="Reported_Fuel_Type_Code",
             y="Annual_Carbon_Intensity_MtCO2_per_MWh",
             text="Annual_Carbon_Intensity_MtCO2_per_MWh")
fig.show()
```



MONTHLY

In [71]:
```
df_Monthly_Carbon_Intensity_by_Fuel = pd.concat([data_2019_workspace_dataframe.iloc[:,[5]].copy(),data_2019_workspace_dataframe
.iloc[:,52:].copy()],axis=1)
```

In [72]:
```
df_Monthly_Carbon_Intensity_by_Fuel=df_Monthly_Carbon_Intensity_by_Fuel.groupby(by=["Reported_Fuel_Type_Code"]).sum()
```

In [73]:
```
df_Monthly_Carbon_Intensity_by_Fuel.reset_index(level=0,inplace=True)
```

In [74]:
```
df_Monthly_Carbon_Intensity_by_Fuel=df_Monthly_Carbon_Intensity_by_Fuel.sort_values(by=["January_Carbon_Intensity_MtCO2_per_MW
h"], ascending=False)
```

In [75]:

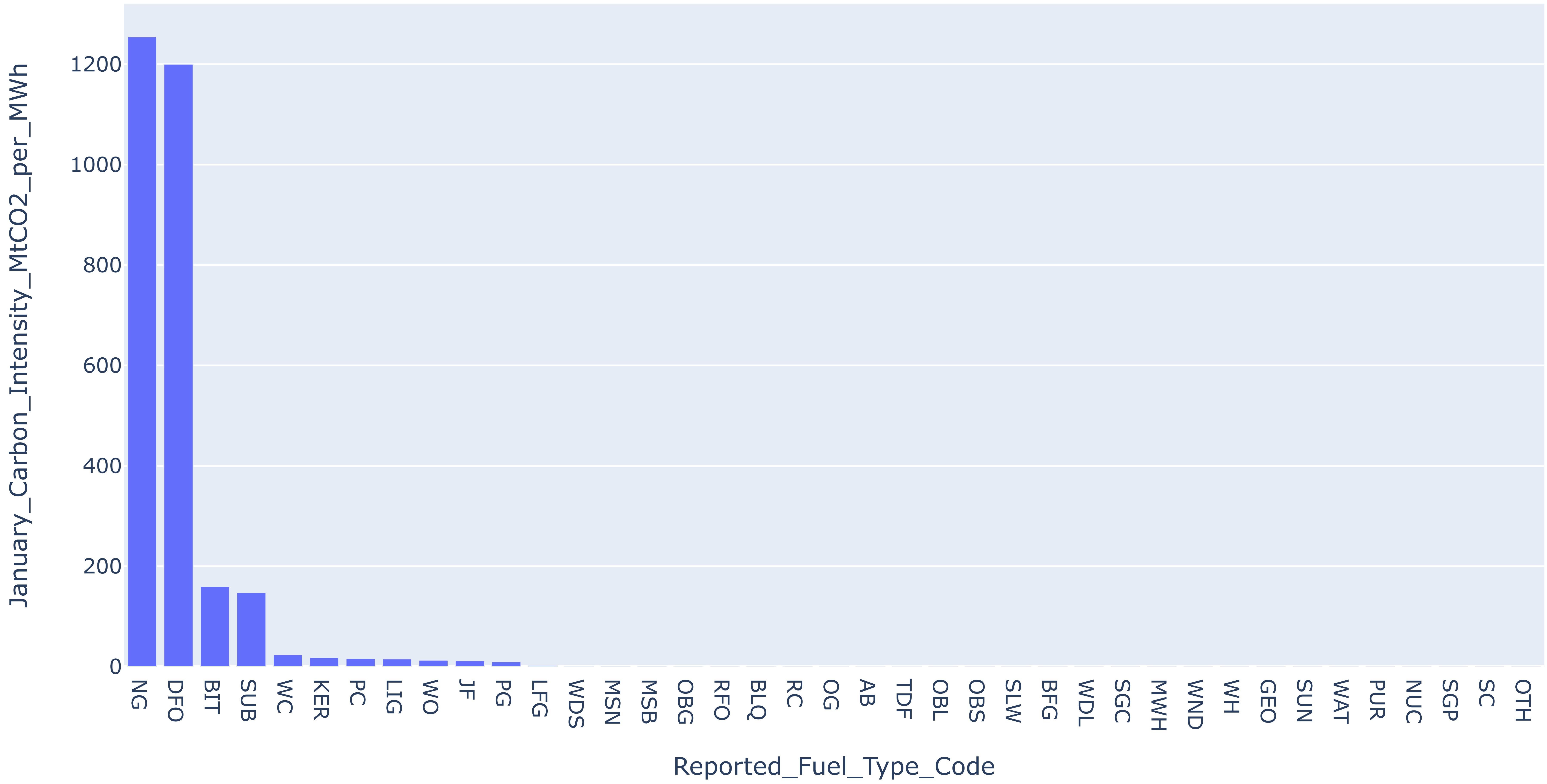
```
df_Monthly_Carbon_Intensity_by_Fuel
```

Out[75]:

	Reported_Fuel_Type_Code	January_Carbon_Intensity_MtCO2_per_MWh	February_Carbon_Intensity_MtCO2_per_MWh	March_Carbon_Intensity_MtCO2_per_MWh
13	NG	1254.627046	1702.491925	1618.635139
4	DFO	1200.017989	1964.571139	2201.836938
2	BIT	159.640930	155.537026	168.701927
29	SUB	147.411211	150.381260	143.097612
33	WC	23.773675	24.174404	21.830477
7	KER	18.050279	18.015990	15.593673
20	PC	16.121385	15.970705	16.113103
9	LIG	15.264872	15.277707	15.840491
38	WO	12.850742	8.673536	9.475856
6	JF	11.874137	13.421581	22.220135
21	PG	9.627269	8.510018	7.023928
8	LFG	2.282593	2.262185	2.276272
35	WDS	0.858763	0.853157	0.827893
11	MSN	0.647819	0.555915	0.587201
10	MSB	0.647815	0.555915	0.587201
15	OBG	0.396007	0.393867	0.402997
24	RFO	0.250343	0.166876	0.147779
3	BLQ	0.234816	0.237597	0.234807
23	RC	0.226323	0.208612	0.223293
18	OG	0.221260	0.211405	0.225065
0	AB	0.220530	0.044735	0.043538
31	TDF	0.162352	0.143753	0.139340
16	OBL	0.049528	0.052355	0.042783
17	OBS	0.045116	0.045454	0.054782
28	SLW	0.033129	0.033137	0.033101
1	BFG	0.028173	0.028861	0.027592
34	WDL	0.007721	0.007721	0.007721
26	SGC	0.006627	0.006395	0.006613

	Reported_Fuel_Type_Code	January_Carbon_Intensity_MtCO2_per_MWh	February_Carbon_Intensity_MtCO2_per_MWh	March_Carbon_Intensity_MtCO2_per_MWh
12	MWH	0.000000	0.000000	0.000000
37	WND	0.000000	0.000000	0.000000
36	WH	0.000000	0.000000	0.000000
5	GEO	0.000000	0.000000	0.000000
30	SUN	0.000000	0.000000	0.000000
32	WAT	0.000000	0.000000	0.000000
22	PUR	0.000000	0.000000	0.000000
14	NUC	0.000000	0.000000	0.000000
27	SGP	0.000000	0.000000	0.000000
25	SC	0.000000	0.000000	0.000000
19	OTH	0.000000	0.000000	0.000000

In [76]:
```
fig = px.bar(df_Monthly_Carbon_Intensity_by_Fuel,
             x="Reported_Fuel_Type_Code",
             y="January_Carbon_Intensity_MtCO2_per_MWh")
fig.show()
```



Carbon Intensity in MtCO2/MWh by State

ANNUALLY

In [77]:
```
df_Annual_Carbon_Intensity_by_State = pd.concat([data_2019_workspace_dataframe.iloc[:,[3,39]].copy()],axis=1)
df_Annual_Carbon_Intensity_by_State=df_Annual_Carbon_Intensity_by_State.groupby(by=["Plant_State"]).sum()
df_Annual_Carbon_Intensity_by_State.reset_index(level=0,inplace=True)
df_Annual_Carbon_Intensity_by_State=df_Annual_Carbon_Intensity_by_State.sort_values(by=["Annual_Carbon_Intensity_MtCO2_per_MWh"
], ascending=False)
```

In [78]:

```
df_Annual_Carbon_Intensity_by_State
```

Out[78]:

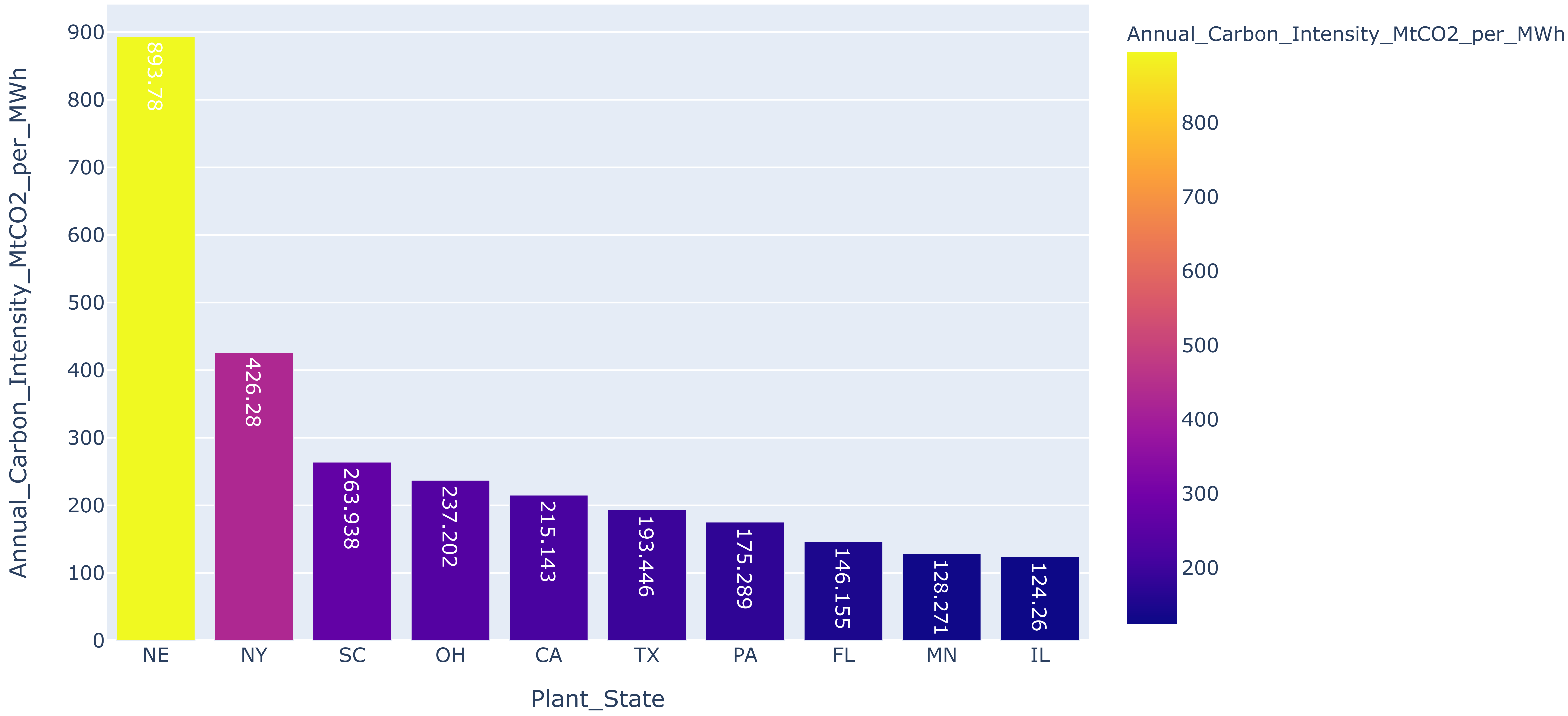
	Plant_State	Annual_Carbon_Intensity_MtCO2_per_MWh
29	NE	893.780456
34	NY	426.279632
40	SC	263.937868
35	OH	237.202191
4	CA	215.143485
43	TX	193.445731
38	PA	175.289334
9	FL	146.154882
23	MN	128.271383
14	IL	124.259968
22	MI	105.506340
24	MO	101.432445
45	VA	100.505690
16	KS	96.414627
10	GA	94.891949
0	AK	85.348284
15	IN	82.258494
27	NC	78.415050
12	IA	74.937313
18	LA	73.060141
19	MA	65.268994
31	NJ	59.996881
5	CO	51.456914
20	MD	46.837319
42	TN	44.028936
3	AZ	43.635254
36	OK	42.898901
1	AL	39.078196

	Plant_State	Annual_Carbon_Intensity_MtCO2_per_MWh
17	KY	34.375446
44	UT	33.627170
25	MS	30.448331
50	WY	29.438939
11	HI	27.404241
2	AR	25.851336
28	ND	25.798392
6	CT	25.050774
8	DE	23.522547
26	MT	23.287317
47	WA	23.167556
32	NM	22.749809
49	WV	19.161363
21	ME	18.811888
41	SD	18.271189
33	NV	17.883420
30	NH	17.221115
37	OR	14.306980
46	VT	14.100240
39	RI	12.164121
13	ID	5.605034
7	DC	2.593406
48	WI	1.071111

In [79]:
```
df_Annual_Carbon_Intensity_by_State_Plotting_data = df_Annual_Carbon_Intensity_by_State.nlargest(10,'Annual_Carbon_Intensity_MtCO2_per_MWh').copy()
```

In [80]:
```
df_Annual_Carbon_Intensity_by_State_Plotting_data["Annual_Carbon_Intensity_MtCO2_per_MWh"]=df_Annual_Carbon_Intensity_by_State_Plotting_data["Annual_Carbon_Intensity_MtCO2_per_MWh"].round(3)
```

In [81]:
```
fig = px.bar(df_Annual_Carbon_Intensity_by_State_Plotting_data,
             x="Plant_State",
             y="Annual_Carbon_Intensity_MtCO2_per_MWh",
             text="Annual_Carbon_Intensity_MtCO2_per_MWh",
             color="Annual_Carbon_Intensity_MtCO2_per_MWh")
fig.show()
```



MONTHLY

In [82]:
```
df_Monthly_Carbon_Intensity_by_State = pd.concat([data_2019_workspace_dataframe.iloc[:,[3]].copy(),data_2019_workspace_datafram
e.iloc[:,52:].copy()],axis=1)
df_Monthly_Carbon_Intensity_by_State=df_Monthly_Carbon_Intensity_by_State.groupby(by=["Plant_State"]).sum()
df_Monthly_Carbon_Intensity_by_State.reset_index(level=0,inplace=True)
df_Monthly_Carbon_Intensity_by_State=df_Monthly_Carbon_Intensity_by_State.sort_values(by=["January_Carbon_Intensity_MtCO2_per_M
Wh"], ascending=False)
```

In [83]:

```
df_Monthly_Carbon_Intensity_by_State
```

Out[83]:

	Plant_State	January_Carbon_Intensity_MtCO2_per_MWh	February_Carbon_Intensity_MtCO2_per_MWh	March_Carbon_Intensity_MtCO2_per_MWh	April_Carbor
29	NE	841.416515	842.743599	847.975435	
4	CA	427.274551	209.968280	191.641887	
34	NY	411.948177	412.450465	519.237782	
40	SC	302.978217	223.865531	242.536379	
35	OH	224.506707	222.279679	223.464867	
43	TX	188.634975	135.538195	150.860359	
9	FL	131.267447	123.143423	121.079548	
38	PA	130.260664	124.740620	116.885031	
14	IL	115.042501	112.735727	121.354350	
23	MN	109.059653	106.648294	93.098524	
22	MI	100.725803	88.805273	91.075175	
45	VA	90.269094	91.548336	124.674334	
24	MO	83.373280	81.390705	86.907046	
0	AK	81.114227	78.881056	78.481325	
15	IN	80.764487	-61.833961	56.037168	
12	IA	79.137166	54.038496	63.087549	
16	KS	79.132948	76.997451	84.697004	
42	TN	68.184725	46.188956	90.511873	
5	CO	66.372696	40.226728	41.717476	
10	GA	65.812117	54.618601	62.827028	
48	WI	65.379756	80.348343	63.061293	
19	MA	53.657686	43.521745	45.986814	
20	MD	46.315812	45.593332	46.766211	
18	LA	42.303515	41.009053	43.616044	
36	OK	41.945927	37.592353	39.749939	
3	AZ	38.876597	37.161772	37.250296	
31	NJ	38.089371	164.522218	47.847439	
17	KY	37.576296	-6.129895	32.448852	

	Plant_State	January_Carbon_Intensity_MtCO2_per_MWh	February_Carbon_Intensity_MtCO2_per_MWh	March_Carbon_Intensity_MtCO2_per_MWh	April_Carbor
1	AL	36.712140	37.622776	34.067024	
41	SD	29.347353	107.407075	16.024625	
50	WY	29.313687	27.273265	28.742864	
11	HI	26.594200	47.659040	24.674685	
44	UT	26.572558	26.021858	25.440997	
25	MS	25.448550	25.435330	26.002340	
6	CT	23.874367	19.021662	20.891283	
2	AR	23.054205	21.278272	19.775263	
28	ND	20.285548	22.389453	15.575959	
47	WA	20.066823	20.014321	20.808533	
32	NM	19.923960	19.842976	19.652293	
26	MT	19.830673	21.384165	18.202632	
8	DE	19.464087	19.199656	11.840235	
49	WV	18.364732	16.619339	17.537904	
21	ME	16.460498	13.781054	15.383822	
33	NV	15.381479	15.645759	16.954791	
37	OR	14.650121	14.061140	18.961007	
39	RI	11.995731	8.741301	13.184653	
30	NH	11.651149	14.076887	13.431484	
46	VT	8.869371	10.726241	10.995279	
13	ID	4.412584	4.440010	5.433007	
7	DC	2.267975	2.549232	2.215452	
27	NC	-1590.384253	59.048040	85.564093	

Carbon Intensity in MtCO2/MWh by Power Plant

ANNUALLY

In [84]:
```
df_Annual_Carbon_Intensity_by_PowerPlant = pd.concat([data_2019_workspace_dataframe.iloc[:,[0,1,39]].copy()],axis=1)
```

In [85]:
```
df_Annual_Carbon_Intensity_by_PowerPlant.head()
```

Out[85]:

	Plant_Id	Plant_Name	Annual_Carbon_Intensity_MtCO2_per_MWh
0	1	Sand Point	0.777661
1	1	Sand Point	0.000000
2	2	Bankhead Dam	-0.000000
3	3	Barry	0.028084
4	3	Barry	0.554370

In [86]:
```
df_Annual_Carbon_Intensity_by_PowerPlant = pd.concat([data_2019_workspace_dataframe.iloc[:,[0,1,39]].copy()],axis=1)
df_Annual_Carbon_Intensity_by_PowerPlant=df_Annual_Carbon_Intensity_by_PowerPlant.groupby(by=["Plant_Name"]).sum()
df_Annual_Carbon_Intensity_by_PowerPlant.reset_index(level=0,inplace=True)
df_Annual_Carbon_Intensity_by_PowerPlant=df_Annual_Carbon_Intensity_by_PowerPlant.sort_values(by=["Annual_Carbon_Intensity_MtCO2_per_MWh"], ascending=False)
```

In [87]:
```
df_Annual_Carbon_Intensity_by_PowerPlant
```

Out[87]:

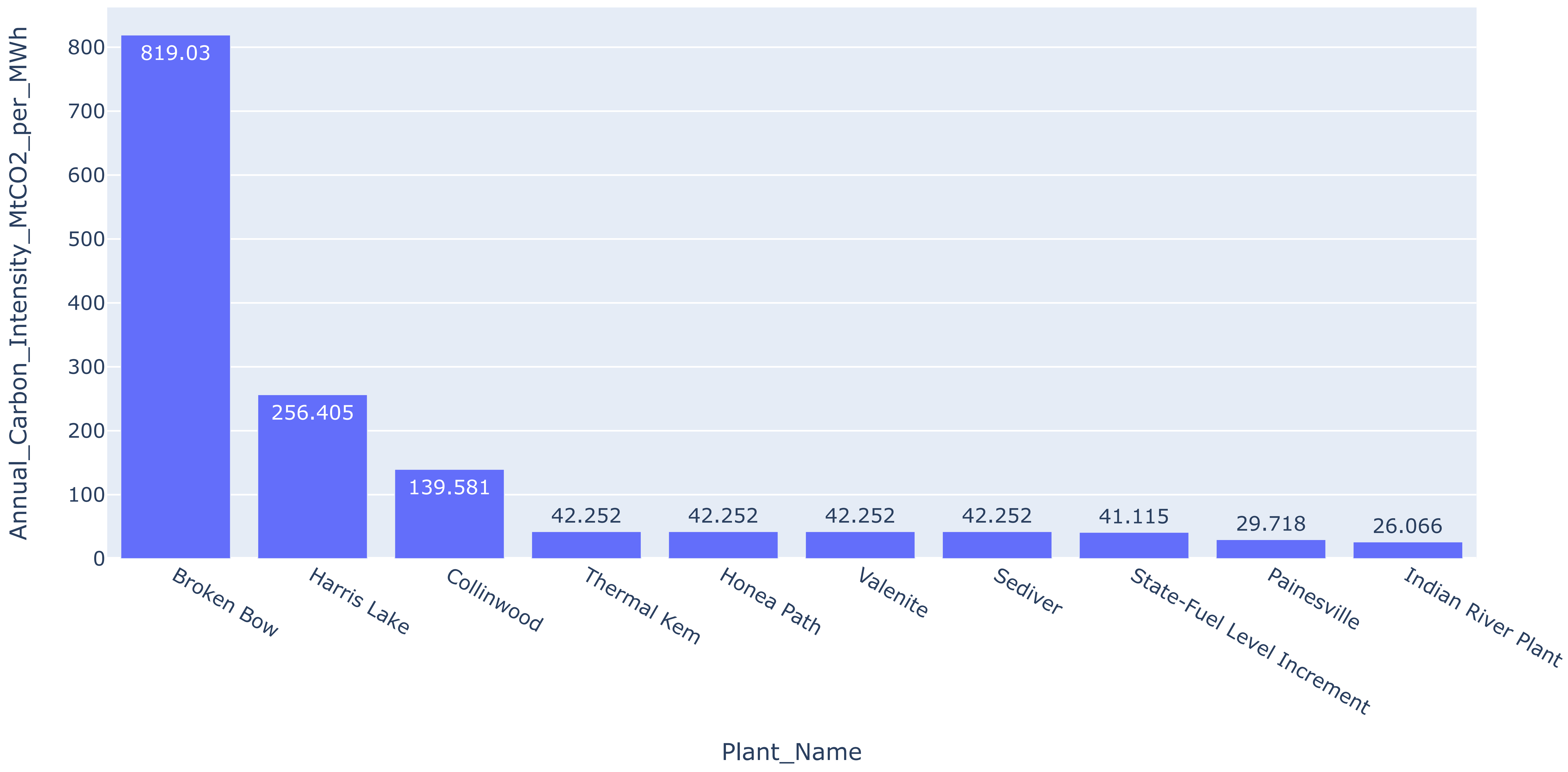
	Plant_Name	Plant_Id	Annual_Carbon_Intensity_MtCO2_per_MWh
1114	Broken Bow	4442	819.030278
3754	Harris Lake	2528	256.405238
1861	Collinwood	5812	139.581111
8603	Thermal Kem	56129	42.252381
3984	Honea Path	56132	42.252381
...	...	...	...
8716	Traer Main	2384	-18.873978
5954	New Prague	3998	-24.795919
6196	Oberlin (OH)	5866	-28.057741
6328	Osage (IA)	4688	-33.753503
3140	French Island	20025	-70.895876

9796 rows × 3 columns

In [88]:
```
df_Annual_Carbon_Intensity_by_PowerPlant_Plotting_data = df_Annual_Carbon_Intensity_by_PowerPlant.nlargest(10,'Annual_Carbon_Intensity_MtCO2_per_MWh')
```

In [89]:
```
df_Annual_Carbon_Intensity_by_PowerPlant_Plotting_data["Annual_Carbon_Intensity_MtCO2_per_MWh"]=df_Annual_Carbon_Intensity_by_PowerPlant_Plotting_data["Annual_Carbon_Intensity_MtCO2_per_MWh"].round(3)
```

In [90]:
```
fig = px.bar(df_Annual_Carbon_Intensity_by_PowerPlant_Plotting_data,
             x="Plant_Name",
             y="Annual_Carbon_Intensity_MtCO2_per_MWh",
             text="Annual_Carbon_Intensity_MtCO2_per_MWh")
fig.show()
```



MONTHLY

In [91]:
```
df_Monthly_Carbon_Intensity_by_PowerPlant = pd.concat([data_2019_workspace_dataframe.iloc[:,[0,1]].copy(),data_2019_workspace_d
ataframe.iloc[:,52:].copy()],axis=1)
df_Monthly_Carbon_Intensity_by_PowerPlant=df_Monthly_Carbon_Intensity_by_PowerPlant.groupby(by=["Plant_Name"]).sum()
df_Monthly_Carbon_Intensity_by_PowerPlant.reset_index(level=0,inplace=True)
df_Monthly_Carbon_Intensity_by_PowerPlant=df_Monthly_Carbon_Intensity_by_PowerPlant.sort_values(by=["January_Carbon_Intensity_M
tCO2_per_MWh"], ascending=False)
```

In [92]:
```
df_Monthly_Carbon_Intensity_by_PowerPlant
```

Out[92]:

	Plant_Name	Plant_Id	January_Carbon_Intensity_MtCO2_per_MWh	February_Carbon_Intensity_MtCO2_per_MWh	March_Carbon_Intensity_MtCO2_per_MWh
1114	Broken Bow	4442	781.265894	781.184284	781.157687
3754	Harris Lake	2528	255.122912	256.744455	407.698413
6319	Ormond Beach	350	247.160454	0.000000	0.000000
1861	Collinwood	5812	136.132575	139.670844	140.271062
9126	W S Lee	22848	64.916225	-16.156735	1.673784
...	...	...	...	...	...
5813	NAEA Lakewood LLC	218560	-19.778319	0.788465	0.711547
5954	New Prague	3998	-23.943802	-24.831119	-24.834083
6196	Oberlin (OH)	5866	-27.538595	-28.141656	-28.192224
6328	Osage (IA)	4688	-35.940139	-33.313235	-33.907800
4931	Lincoln Combustion	14554	-1648.622129	-1.407010	9.678929

9796 rows × 14 columns

In [92]:

2019 - 2009 Data

In [93]:
```
data_2018 = pd.read_excel("/content/drive/MyDrive/Colab Notebooks/ElectricityCO2Emission/AllData/2018/EIA923_Schedules_2_3_4_5_M_12_2018_Final_Revision.xlsx", sheet_name="Page 1 Generation and Fuel Data",skiprows=5)
```

In [94]:
```
data_2017 = pd.read_excel("/content/drive/MyDrive/Colab Notebooks/ElectricityCO2Emission/AllData/2017/EIA923_Schedules_2_3_4_5_M_12_2017_Final_Revision.xlsx", sheet_name="Page 1 Generation and Fuel Data",skiprows=5)
```

In [95]:
```
data_2016 = pd.read_excel("/content/drive/MyDrive/Colab Notebooks/ElectricityCO2Emission/AllData/2016/EIA923_Schedules_2_3_4_5_M_12_2016_Final_Revision.xlsx", sheet_name="Page 1 Generation and Fuel Data",skiprows=5)
```

In [96]:
```
data_2015 = pd.read_excel("/content/drive/MyDrive/Colab Notebooks/ElectricityCO2Emission/AllData/2015/EIA923_Schedules_2_3_4_5_M_12_2015_Final_Revision.xlsx", sheet_name="Page 1 Generation and Fuel Data",skiprows=5)
```

In [97]:
```
data_2014 = pd.read_excel("/content/drive/MyDrive/Colab Notebooks/ElectricityCO2Emission/AllData/2014/EIA923_Schedules_2_3_4_5_M_12_2014_Final_Revision.xlsx", sheet_name="Page 1 Generation and Fuel Data",skiprows=5)
```

In [98]:
```
data_2013 = pd.read_excel("/content/drive/MyDrive/Colab Notebooks/ElectricityCO2Emission/AllData/2013/EIA923_Schedules_2_3_4_5_2013_Final_Revision.xlsx", sheet_name="Page 1 Generation and Fuel Data",skiprows=5)
```

In [99]:
```
data_2012 = pd.read_excel("/content/drive/MyDrive/Colab Notebooks/ElectricityCO2Emission/AllData/2012/EIA923_Schedules_2_3_4_5_M_12_2012_Final_Revision.xlsx", sheet_name="Page 1 Generation and Fuel Data",skiprows=5)
```

In [100]:
```
data_2011 = pd.read_excel("/content/drive/MyDrive/Colab Notebooks/ElectricityCO2Emission/AllData/2011/EIA923_Schedules_2_3_4_5_2011_Final_Revision.xlsx", sheet_name="Page 1 Generation and Fuel Data",skiprows=5)
```

In [101]:
```
data_2010 = pd.read_excel("/content/drive/MyDrive/Colab Notebooks/ElectricityCO2Emission/AllData/2010/EIA923 SCHEDULES 2_3_4_5 Final 2010.xls", sheet_name="Page 1 Generation and Fuel Data",skiprows=7)
```

In [102]:
```
data_2009 = pd.read_excel("/content/drive/MyDrive/Colab Notebooks/ElectricityCO2Emission/AllData/2009/EIA923 SCHEDULES 2_3_4_5 M Final 2009 REVISED 05252011.XLS", sheet_name="Page 1 Generation and Fuel Data",skiprows=7)
```

In [103]:
```
data_2019_dataframe = pd.concat([data_2019.iloc[:,[14,18]].copy(),data_2019.iloc[:,94:].copy()],axis=1)
data_2018_dataframe = pd.concat([data_2018.iloc[:,[14,18]].copy(),data_2018.iloc[:,94:].copy()],axis=1)
data_2017_dataframe = pd.concat([data_2017.iloc[:,[14,18]].copy(),data_2017.iloc[:,94:].copy()],axis=1)
data_2016_dataframe = pd.concat([data_2016.iloc[:,[14,18]].copy(),data_2016.iloc[:,94:].copy()],axis=1)
data_2015_dataframe = pd.concat([data_2015.iloc[:,[14,18]].copy(),data_2015.iloc[:,94:].copy()],axis=1)
data_2014_dataframe = pd.concat([data_2014.iloc[:,[14,18]].copy(),data_2014.iloc[:,94:].copy()],axis=1)
data_2013_dataframe = pd.concat([data_2013.iloc[:,[14,18]].copy(),data_2013.iloc[:,94:].copy()],axis=1)
data_2012_dataframe = pd.concat([data_2012.iloc[:,[14,18]].copy(),data_2012.iloc[:,94:].copy()],axis=1)
data_2011_dataframe = pd.concat([data_2011.iloc[:,[14,18]].copy(),data_2011.iloc[:,94:].copy()],axis=1)
data_2010_dataframe = pd.concat([data_2010.iloc[:,[14,18]].copy(),data_2010.iloc[:,94:].copy()],axis=1)
data_2009_dataframe = pd.concat([data_2009.iloc[:,[14,18]].copy(),data_2009.iloc[:,94:].copy()],axis=1)
```

In [104]:
```
dataframes = [data_2009_dataframe,data_2010_dataframe,data_2011_dataframe,data_2012_dataframe,data_2013_dataframe,data_2014_dat
aframe,data_2015_dataframe,data_2016_dataframe,data_2017_dataframe,data_2018_dataframe,data_2019_dataframe]
```

In [105]:
```
for df in dataframes:
    for c in df.columns:
        new_column_name = (c.replace("\n","_")).replace(" ","_").lower()
        df.rename(columns={c:new_column_name},inplace=True)
```

In [106]:
```
data_2009_dataframe.rename(columns={"elec_fuel_consumption_mmbtus":"elec_fuel_consumption_mmbtu"},inplace=True)
data_2010_dataframe.rename(columns={"elec_fuel_consumption_mmbtus":"elec_fuel_consumption_mmbtu"},inplace=True)
```

In [107]:
```
final_dataframe = pd.concat(dataframes)
```

In [108]:
```
final_dataframe
```

Out[108]:

	reported_fuel_type_code	physical_unit_label	elec_fuel_consumption_mmbtu	net_generation_(megawatthours)	year
0	WAT	NaN	2758750.0	282659.000	2009
1	NG	mcf	313469.0	2156430.000	2009
2	NG	mcf	41903740.0	3888492.000	2009
3	BIT	short tons	79317545.0	7947703.984	2009
4	DFO	barrels	0.0	0.000	2009
...	...	...	...	...	...
14512	WDS	short tons	0.0	0.000	2019
14513	WO	barrels	8938.0	849.573	2019
14514	WND	NaN	36377.0	4084.943	2019
14515	WND	NaN	109220.0	12264.976	2019
14516	WND	NaN	709476.0	79671.592	2019

132312 rows × 5 columns

In [109]:
```
fuel_data = CO2_fuel_data_2019.copy()
```

In [110]:
```
for c in fuel_data.columns:
    new_column_name = c.lower()
    fuel_data.rename(columns={c:new_column_name},inplace=True)
```

In [111]:

```
fuel_data
```

Out[111]:

	reported_fuel_type_code	description	co2_emission_pound_per_mmbtu	mapped_fuel_name
0	AB	agricultural by-products	1.00	
1	ANT	anthracite coal	228.60	anthracite
2	BFG	blast furnace gas	1.00	
3	BIT	bituminous coal	205.40	bituminous
4	BLQ	black liquor	1.00	
5	DFO	distillate fuel oil. including diesel, no. 1, ...	163.45	diesel
6	GEO	geothermal	1.00	
7	JF	jet fuel	159.25	jet fuel
8	KER	kerosene	161.35	kerosene
9	LFG	landfill gas	1.00	
10	LIG	lignite coal	216.24	lignite
11	MSB	biogenic municipal solid waste	1.00	
12	MSN	non-biogenic municipal solid waste	1.00	
13	MWH	electricity used for energy storage	1.00	
14	NG	natural gas	116.65	natural gas
15	NUC	nuclear. including uranium, plutonium, and tho...	1.00	
16	OBG	other biomass gas. including digester gas, met...	1.00	
17	OBL	other biomass liquids	1.00	
18	OBS	other biomass solids	1.00	
19	OG	other gas	1.00	
20	OTH	other fuel	1.00	
21	PC	petroleum coke	250.59	coke
22	PG	gaseous propane	138.63	propane
23	PUR	purchased steam	1.00	
24	RC	refined coal	1.00	
25	RFO	residual fuel oil. including no. 5 & 6 fuel oi...	1.00	
26	SC	coal-based synfuel. including briquettes, pell...	1.00	
27	SGC	coal-derived synthesis gas	1.00	

	reported_fuel_type_code	description	co2_emission_pound_per_mmbtu	mapped_fuel_name
28	SGP	synthesis gas from petroleum coke	250.59	coke
29	SLW	sludge waste	1.00	
30	SUB	subbituminous coal	214.13	subbituminous
31	SUN	solar	1.00	
32	TDF	tire-derived fuels	1.00	
33	WAT	water at a conventional hydroelectric turbine ...	1.00	
34	WC	waste/other coal. including anthracite culm, b...	216.24	lignite
35	WDL	wood waste liquids, excluding black liquor. in...	1.00	
36	WDS	wood/wood waste solids. including paper pellet...	1.00	
37	WH	waste heat not directly attributed to a fuel s...	1.00	
38	WND	wind	1.00	
39	WO	waste/other oil. including crude oil, liquid b...	138.63	propane

In [112]:
```
final_dataframe = pd.merge(final_dataframe,fuel_data,how='left',on=["reported_fuel_type_code"])
```

In [113]:
```
final_dataframe.head()
```

Out[113]:

	reported_fuel_type_code	physical_unit_label	elec_fuel_consumption_mmbtu	net_generation_(megawatthours)	year	description	co2_emission_pound_per_m
0	WAT	NaN	2758750.0	282659.000	2009	water at a conventional hydroelectric turbine ...	
1	NG	mcf	313469.0	2156430.000	2009	natural gas	1
2	NG	mcf	41903740.0	3888492.000	2009	natural gas	1
3	BIT	short tons	79317545.0	7947703.984	2009	bituminous coal	2
4	DFO	barrels	0.0	0.000	2009	distillate fuel oil. including diesel, no. 1, ...	1

In [114]:
```
final_dataframe["physical_unit_label"] = final_dataframe["physical_unit_label"].astype('str')
```

In [115]:
```
final_dataframe.loc[(final_dataframe["physical_unit_label"] == "nan"),"co2_emission_pound_per_mmbtu"]=0
```

In [116]:
```
# 1 MMBtu of 'Natural Gas' emits 116.65 pounds of CO2
# Find Mt of CO2 emitted for the Elec_Fuel_Consumption_MMBtu
final_dataframe["annual_elect_CO2_emission_MtCO2"] = (final_dataframe["elec_fuel_consumption_mmbtu"]*final_dataframe["co2_emiss
ion_pound_per_mmbtu"])/2205
```

In [117]:
```
# Carbon Intensity = CO2 emitted per unit of electric energy generated
# Carbon Intensity = (Total amount of electricity related CO2 emitted in Metric tons) / (Total amount of electricity produced i
n Megawatthours) = Carbon Intensity in MtCO2/MWh

# using fillna to handle 0/0 giving NaN
final_dataframe["annual_carbon_intensity_MtCO2_per_MWh"] = (final_dataframe["annual_elect_CO2_emission_MtCO2"]/final_dataframe[
"net_generation_(megawatthours)"]).fillna(0)

# if above there was a (something)/0, then it gives inf so replace inf with 0
final_dataframe.loc[final_dataframe["annual_carbon_intensity_MtCO2_per_MWh"] == np.inf, "annual_carbon_intensity_MtCO2_per_MWh"
] = 0
final_dataframe.loc[final_dataframe["annual_carbon_intensity_MtCO2_per_MWh"] == -np.inf, "annual_carbon_intensity_MtCO2_per_MW
h"] = 0
```

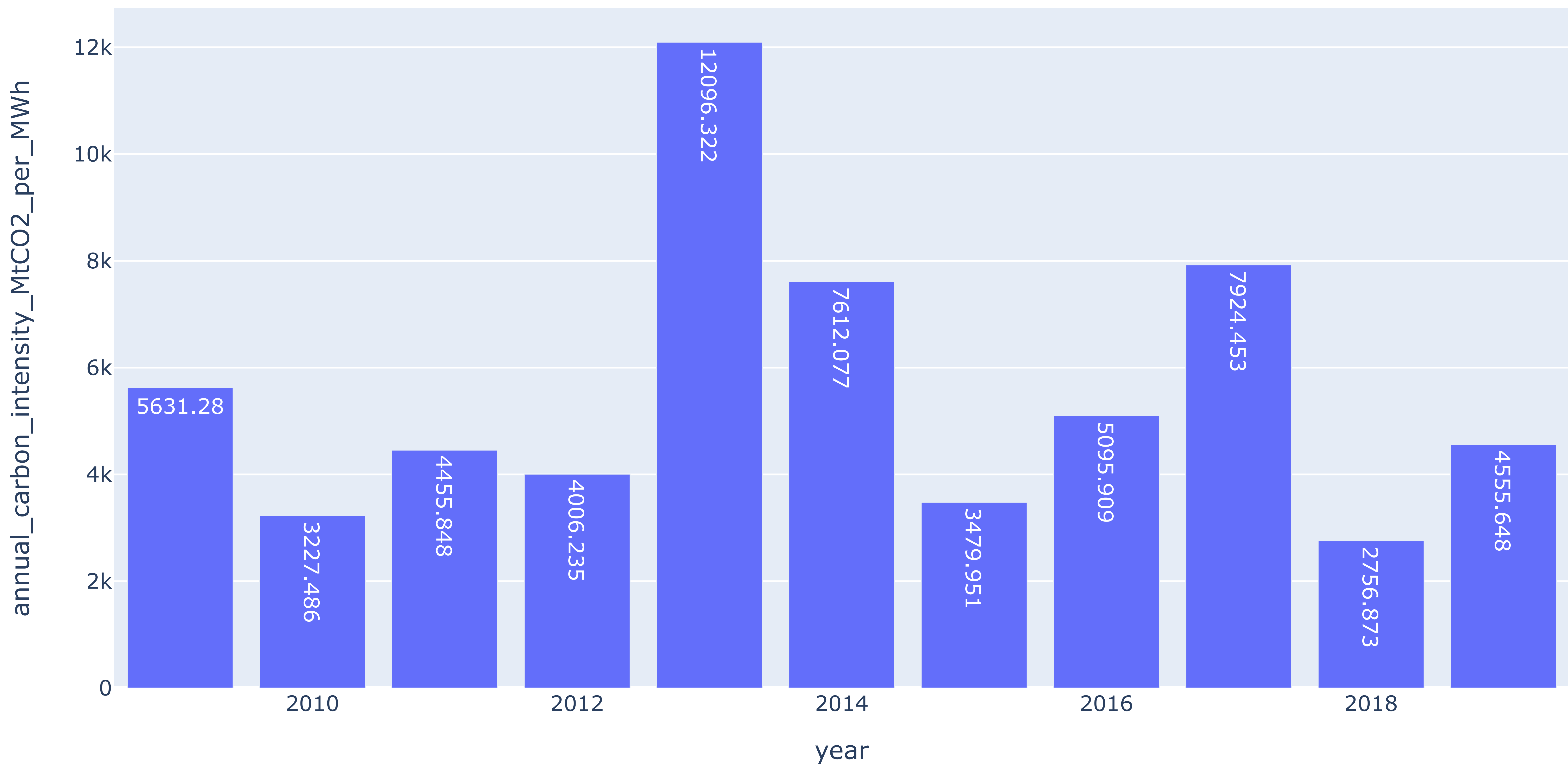
In [118]:
```
df_Annual_Carbon_Intensity_by_Year = pd.concat([final_dataframe.iloc[:,[4,9]].copy()],axis=1)
```

In [119]:
```
df_Annual_Carbon_Intensity_by_Year=df_Annual_Carbon_Intensity_by_Year.groupby(by=["year"]).sum()
```

In [120]:
```
df_Annual_Carbon_Intensity_by_Year.reset_index(level=0,inplace=True)
```

In [121]:
```
df_Annual_Carbon_Intensity_by_Year["annual_carbon_intensity_MtCO2_per_MWh"] = df_Annual_Carbon_Intensity_by_Year["annual_carbon
_intensity_MtCO2_per_MWh"].round(3)
```

In [122]:
```
fig = px.bar(df_Annual_Carbon_Intensity_by_Year, y="annual_carbon_intensity_MtCO2_per_MWh", x="year", text="annual_carbon_inten
sity_MtCO2_per_MWh")
fig.show()
```



Note: The yearly data shows 2013 being the year with the highest CO2 emission. More digging can be done to check the cause of this excess power generation. Or maybe one particular fuel / group of fuels with a high CO2 emission value caused this.

In [126]:
```
!jupyter nbconvert — to html ElectricityCO2emission.ipynb
```

```
[NbConvertApp] WARNING | pattern u'\u2014' matched no files
[NbConvertApp] WARNING | pattern u'to' matched no files
[NbConvertApp] WARNING | pattern u'html' matched no files
[NbConvertApp] Converting notebook ElectricityCO2emission.ipynb to html
[NbConvertApp] Writing 646266 bytes to ElectricityCO2emission.html
```

In [124]:
```
# Restore the notebook mode
plotly.io.renderers.default = 'colab'
```

In [124]:
```
```