GitHub

https://github.com/lizzy-miller/Lab3

a

In [1]:
```
import pandas as pd
import numpy as np
import requests
import os
import psycopg2
import zipfile
import io
from sqlalchemy import create_engine
```

In [2]:
```
POSTGRES_PASSWORD = os.getenv('POSTGRES_PASSWORD')
```

b

In [3]:
```
url ='https://databank.worldbank.org/data/download/ESG_CSV.zip'
r = requests.get(url)
z = zipfile.ZipFile(io.BytesIO(r.content))
z.extractall()
```

In [4]:
```
url = 'https://v-dem.net/media/datasets/V-Dem-CY-Core_csv_v13.zip'
r = requests.get(url)
z = zipfile.ZipFile(io.BytesIO(r.content))
z.extractall()
```

In [5]:
```
vdem = pd.read_csv('V-Dem-CY-Core-v13.csv')
wb = pd.read_csv('ESGData.csv')
country = pd.read_csv('ESGCountry.csv')
```

C. V-Dem

In [6]:
```
vdem_clean = vdem[['country_text_id', 'country_name', 'year', 'v2x_polyarchy
```

In [7]:
```
vdem_clean = vdem_clean.query("year >= 1960 and year <= 2021")
```

In [8]:
```
vdem_clean = vdem_clean.rename({'country_text_id': 'country_code',
                                'country_name': 'country_name_vdem',
                                'v2x_polyarchy':'democracy'}, axis = 1)
```

In [9]:
```
vdem_clean = vdem_clean.sort_values(by = ['country_code', 'year'])
```

In [10]:
```
vdem_clean.head(10)
```

Out[10]:

	country_code	country_name_vdem	year	democracy
5433	AFG	Afghanistan	1960	0.080
5434	AFG	Afghanistan	1961	0.083
5435	AFG	Afghanistan	1962	0.082
5436	AFG	Afghanistan	1963	0.085
5437	AFG	Afghanistan	1964	0.137
5438	AFG	Afghanistan	1965	0.150
5439	AFG	Afghanistan	1966	0.161
5440	AFG	Afghanistan	1967	0.163
5441	AFG	Afghanistan	1968	0.163
5442	AFG	Afghanistan	1969	0.162

D. ESG Country

In [11]:
```
country_clean = country[['Country Code', 'Table Name', 'Long Name', 'Currenc
```

In [12]:
```
country_clean = country_clean.rename({'Country Code': 'country_code',
                                      'Table Name': 'country_name_wb',
                                      'Long Name': 'country_longname',
                                       'Currency Unit': 'currency_unit',
                                       'Region': 'region',
                                       'Income Group': 'income_group'}, axis
```

In [13]:
```
noncountries = ["Arab World", "Central Europe and the Baltics",
 "Caribbean small states",
 "East Asia & Pacific (excluding high income)",
 "Early-demographic dividend","East Asia & Pacific",
 "Europe & Central Asia (excluding high income)",
 "Europe & Central Asia", "Euro area",
 "European Union","Fragile and conflict affected situations",
 "High income",
 "Heavily indebted poor countries (HIPC)","IBRD only",
 "IDA & IBRD total",
 "IDA total","IDA blend","IDA only",
 "Latin America & Caribbean (excluding high income)",
 "Latin America & Caribbean",
 "Least developed countries: UN classification",
 "Low income","Lower middle income","Low & middle income",
 "Late-demographic dividend","Middle East & North Africa",
 "Middle income",
 "Middle East & North Africa (excluding high income)",
 "North America","OECD members",
 "Other small states","Pre-demographic dividend",
 "Pacific island small states",
 "Post-demographic dividend",
 "Sub-Saharan Africa (excluding high income)",
```

```
                "Sub-Saharan Africa",
"Small states","East Asia & Pacific (IDA & IBRD)",
"Europe & Central Asia (IDA & IBRD)",
"Latin America & Caribbean (IDA & IBRD)",
"Middle East & North Africa (IDA & IBRD)","South Asia",
"South Asia (IDA & IBRD)",
"Sub-Saharan Africa (IDA & IBRD)",
"Upper middle income", "World"]
```

In [14]:
```
country_clean = country_clean.query('country_name_wb not in @noncountries')
```

In [15]:
```
country_clean.head(10)
```

Out[15]:

	country_code	country_name_wb	country_longname	currency_unit	region	inc
0	AFG	Afghanistan	Islamic State of Afghanistan	Afghan afghani	South Asia	
1	AGO	Angola	People's Republic of Angola	Angolan kwanza	Sub-Saharan Africa	L
2	ALB	Albania	Republic of Albania	Albanian lek	Europe & Central Asia	U
3	AND	Andorra	Principality of Andorra	Euro	Europe & Central Asia	
5	ARE	United Arab Emirates	United Arab Emirates	U.A.E. dirham	Middle East & North Africa	
6	ARG	Argentina	Argentine Republic	Argentine peso	Latin America & Caribbean	U
7	ARM	Armenia	Republic of Armenia	Armenian dram	Europe & Central Asia	U
8	ATG	Antigua and Barbuda	Antigua and Barbuda	East Caribbean dollar	Latin America & Caribbean	
9	AUS	Australia	Commonwealth of Australia	Australian dollar	East Asia & Pacific	
10	AUT	Austria	Republic of Austria	Euro	Europe & Central Asia	

E. World Bank ESG Data

In [16]:
```
wb_clean = wb[['Country Code', 'Country Name', 'Indicator Code'] + [col for
```

In [17]:
```
wb_clean = wb_clean.rename({'Country Code': 'country_code',
                            'Country Name': 'country_name_wb',
                            'Indicator Code': 'feature'}, axis = 1)
```

In [18]:
```
noncountries.remove('World')
```

In [19]:
```
wb_clean = wb_clean.query('country_name_wb not in @noncountries')
```

In [20]:
```
replace_map = {
 "AG.LND.AGRI.ZS": "agricultural_land",
 "AG.LND.FRST.ZS": "forest_area",
 "AG.PRD.FOOD.XD": "food_production_index",
 "CC.EST": "control_of_corruption",
 "EG.CFT.ACCS.ZS": "access_to_clean_fuels_and_technologies_for_cooking",
 "EG.EGY.PRIM.PP.KD": "energy_intensity_level_of_primary_energy",
 "EG.ELC.ACCS.ZS": "access_to_electricity",
 "EG.ELC.COAL.ZS": "electricity_production_from_coal_sources",
 "EG.ELC.RNEW.ZS": "renewable_electricity_output",
 "EG.FEC.RNEW.ZS": "renewable_energy_consumption",
 "EG.IMP.CONS.ZS": "energy_imports",
 "EG.USE.COMM.FO.ZS": "fossil_fuel_energy_consumption",
 "EG.USE.PCAP.KG.OE": "energy_use",
 "EN.ATM.CO2E.PC": "co2_emissions",
 "EN.ATM.METH.PC": "methane_emissions",
 "EN.ATM.NOXE.PC": "nitrous_oxide_emissions",
 "EN.ATM.PM25.MC.M3": "pm2_5_air_pollution",
 "EN.CLC.CDDY.XD": "cooling_degree_days",
 "EN.CLC.GHGR.MT.CE": "ghg_net_emissions",
 "EN.CLC.HEAT.XD": "heat_index_35",
 "EN.CLC.MDAT.ZS": "droughts",
 "EN.CLC.PRCP.XD": "maximum_5-day_rainfall",
 "EN.CLC.SPEI.XD": "mean_drought_index","EN.MAM.THRD.NO": "mammal_species",
 "EN.POP.DNST": "population_density",
 "ER.H2O.FWTL.ZS": "annual_freshwater_withdrawals",
 "ER.PTD.TOTL.ZS": "terrestrial_and_marine_protected_areas",
 "GB.XPD.RSDV.GD.ZS": "research_and_development_expenditure",
 "GE.EST": "government_effectiveness",
 "IC.BUS.EASE.XQ": "ease_of_doing_business_rank",
 "IC.LGL.CRED.XQ": "strength_of_legal_rights_index",
 "IP.JRN.ARTC.SC": "scientific_and_technical_journal_articles",
 "IP.PAT.RESD": "patent_applications",
 "IT.NET.USER.ZS": "individuals_using_the_internet",
 "NV.AGR.TOTL.ZS": "agriculture",
 "NY.ADJ.DFOR.GN.ZS": "net_forest_depletion",
 "NY.ADJ.DRES.GN.ZS": "natural_resources_depletion",
 "NY.GDP.MKTP.KD.ZG": "gdp_growth",
 "PV.EST": "political_stability_and_absence_of_violence",
 "RL.EST": "rule_of_law",
 "RQ.EST": "regulatory_quality",
```

```
    "SE.ADT.LITR.ZS": "literacy_rate",
    "SE.ENR.PRSC.FM.ZS": "gross_school_enrollment",
    "SE.PRM.ENRR": "primary_school_enrollment",
    "SE.XPD.TOTL.GB.ZS": "government_expenditure_on_education",
    "SG.GEN.PARL.ZS": "proportion_of_seats_held_by_women_in_national_parliament
    "SH.DTH.COMM.ZS": "cause_of_death",
    "SH.DYN.MORT": "mortality_rate",
    "SH.H2O.SMDW.ZS": "people_using_safely_managed_drinking_water_services",
    "SH.MED.BEDS.ZS": "hospital_beds",
    "SH.STA.OWAD.ZS": "prevalence_of_overweight",
    "SH.STA.SMSS.ZS": "people_using_safely_managed_sanitation_services",
    "SI.DST.FRST.20": "income_share_held_by_lowest_20pct",
    "SI.POV.GINI": "gini_index",
    "SI.POV.NAHC": "poverty_headcount_ratio_at_national_poverty_lines",
    "SI.SPR.PCAP.ZG": "annualized_average_growth_rate_in_per_capita_real_surve"
    "SL.TLF.0714.ZS": "children_in_employment",
    "SL.TLF.ACTI.ZS": "labor_force_participation_rate",
    "SL.TLF.CACT.FM.ZS": "ratio_of_female_to_male_labor_force_participation_ra"
    "SL.UEM.TOTL.ZS": "unemployment",
    "SM.POP.NETM": "net_migration",
    "SN.ITK.DEFC.ZS": "prevalence_of_undernourishment",
    "SP.DYN.LE00.IN": "life_expectancy_at_birth",
    "SP.DYN.TFRT.IN": "fertility_rate",
    "SP.POP.65UP.TO.ZS": "population_ages_65_and_above",
    "SP.UWT.TFRT": "unmet_need_for_contraception",
    "VA.EST": "voice_and_accountability",
    "EN.CLC.CSTP.ZS": "coastal_protection",
    "SD.ESR.PERF.XQ": "economic_and_social_rights_performance_score",
    "EN.CLC.HDDY.XD": "heating_degree_days",
    "EN.LND.LTMP.DC": "land_surface_temperature",
    "ER.H2O.FWST.ZS": "freshwater_withdrawal",
    "EN.H2O.BDYS.ZS": "water_quality",
    "AG.LND.FRLS.HA": "tree_cover_loss",
}
```

In [21]:
```
wb_clean['feature'] = wb_clean['feature'].map(replace_map)
```

In [22]:
```
wb_clean = pd.melt(wb_clean, id_vars = ['country_code', 'country_name_wb', '
```

In [23]:
```
wb_clean = wb_clean.rename({'variable' : 'year'}, axis = 1)
```

In [24]:
```
wb_clean = wb_clean.pivot(index=['country_code', 'country_name_wb', 'year'],
                          columns='feature', values='value').reset_i
```

G

In [25]:
```
wb_clean['year'] = wb_clean['year'].astype(int)
wb_clean
```

Out[25]:

feature	country_code	country_name_wb	year	access_to_clean_fuels_and_technolog
0	AFG	Afghanistan	1960	
1	AFG	Afghanistan	1961	
2	AFG	Afghanistan	1962	
3	AFG	Afghanistan	1963	
4	AFG	Afghanistan	1964	
...	...	...	...	
12217	ZWE	Zimbabwe	2018	
12218	ZWE	Zimbabwe	2019	
12219	ZWE	Zimbabwe	2020	
12220	ZWE	Zimbabwe	2021	
12221	ZWE	Zimbabwe	2022	

12222 rows × 74 columns

H.

In [26]:
```
whole_world_data = wb_clean[wb_clean['country_name_wb'] == 'World'].copy()
```

In [27]:
```
whole_world_data.head(10).T
```

Out[27]:

feature	11844	11845	11846	11847	11
country_code	WLD	WLD	WLD	WLD	\
country_name_wb	World	World	World	World	W
year	1960	1961	1962	1963	1
access_to_clean_fuels_and_technologies_for_cooking	NaN	NaN	NaN	NaN	
access_to_electricity	NaN	NaN	NaN	NaN	
...	...	...	...	...	
tree_cover_loss	NaN	NaN	NaN	NaN	
unemployment	NaN	NaN	NaN	NaN	
unmet_need_for_contraception	NaN	NaN	NaN	NaN	
voice_and_accountability	NaN	NaN	NaN	NaN	
water_quality	NaN	NaN	NaN	NaN	

74 rows × 10 columns

In [28]:

```
wb_clean = wb_clean.query("country_name_wb != 'World'")
wb_clean.head(10)
```

Out[28]:

feature	country_code	country_name_wb	year	access_to_clean_fuels_and_technologi
0	AFG	Afghanistan	1960	
1	AFG	Afghanistan	1961	
2	AFG	Afghanistan	1962	
3	AFG	Afghanistan	1963	
4	AFG	Afghanistan	1964	
5	AFG	Afghanistan	1965	
6	AFG	Afghanistan	1966	
7	AFG	Afghanistan	1967	
8	AFG	Afghanistan	1968	
9	AFG	Afghanistan	1969	

10 rows × 74 columns

In [29]:

```
whole_world_data = whole_world_data.drop(['country_code', 'country_name_wb']
```

In [30]:
```
whole_world_data.columns
```

Out[30]:
```
Index(['year', 'access_to_clean_fuels_and_technologies_for_cooking',
       'access_to_electricity', 'agricultural_land', 'agriculture',
       'annual_freshwater_withdrawals',
       'annualized_average_growth_rate_in_per_capita_real_surve',
       'cause_of_death', 'children_in_employment', 'co2_emissions',
       'coastal_protection', 'control_of_corruption', 'cooling_degree_day
s',
       'economic_and_social_rights_performance_score',
       'electricity_production_from_coal_sources', 'energy_imports',
       'energy_intensity_level_of_primary_energy', 'energy_use',
       'fertility_rate', 'food_production_index', 'forest_area',
       'fossil_fuel_energy_consumption', 'freshwater_withdrawal', 'gdp_grow
th',
       'ghg_net_emissions', 'gini_index', 'government_effectiveness',
       'government_expenditure_on_education', 'gross_school_enrollment',
       'heat_index_35', 'heating_degree_days', 'hospital_beds',
       'income_share_held_by_lowest_20pct', 'individuals_using_the_interne
t',
       'labor_force_participation_rate', 'land_surface_temperature',
       'life_expectancy_at_birth', 'literacy_rate', 'mammal_species',
       'mean_drought_index', 'methane_emissions', 'mortality_rate',
       'natural_resources_depletion', 'net_forest_depletion', 'net_migratio
n',
       'nitrous_oxide_emissions', 'patent_applications',
       'people_using_safely_managed_drinking_water_services',
       'people_using_safely_managed_sanitation_services',
       'pm2_5_air_pollution', 'political_stability_and_absence_of_violenc
e',
       'population_ages_65_and_above', 'population_density',
       'poverty_headcount_ratio_at_national_poverty_lines',
       'prevalence_of_overweight', 'prevalence_of_undernourishment',
       'primary_school_enrollment',
       'proportion_of_seats_held_by_women_in_national_parliament',
       'ratio_of_female_to_male_labor_force_participation_ra',
       'regulatory_quality', 'renewable_electricity_output',
       'renewable_energy_consumption', 'research_and_development_expenditur
e',
       'rule_of_law', 'scientific_and_technical_journal_articles',
       'strength_of_legal_rights_index',
       'terrestrial_and_marine_protected_areas', 'tree_cover_loss',
       'unemployment', 'unmet_need_for_contraception',
       'voice_and_accountability', 'water_quality'],
      dtype='object', name='feature')
```

In [31]:
```
new_column_names = {col: f"world_{col}" for col in whole_world_data.columns
whole_world_data_clean = whole_world_data.rename(columns=new_column_names)
whole_world_data_clean['year'] = whole_world_data_clean['year'].astype(int)
whole_world_data_clean
```

Out[31]:

feature	year	world_access_to_clean_fuels_and_technologies_for_cooking	world_acce
11844	1960	NaN	
11845	1961	NaN	
11846	1962	NaN	
11847	1963	NaN	
11848	1964	NaN	
...	...	...	
11902	2018	67.696517	
11903	2019	68.921601	
11904	2020	70.184805	
11905	2021	71.331487	
11906	2022	NaN	

63 rows × 72 columns

i

In [32]:
```
vdem_clean
```

Out[32]:

	country_code	country_name_vdem	year	democracy
5433	AFG	Afghanistan	1960	0.080
5434	AFG	Afghanistan	1961	0.083
5435	AFG	Afghanistan	1962	0.082
5436	AFG	Afghanistan	1963	0.085
5437	AFG	Afghanistan	1964	0.137
...	...	...	...	...
26150	ZZB	Zanzibar	2017	0.267
26151	ZZB	Zanzibar	2018	0.268
26152	ZZB	Zanzibar	2019	0.266
26153	ZZB	Zanzibar	2020	0.258
26154	ZZB	Zanzibar	2021	0.276

10371 rows × 4 columns

In [33]:
```
wb_clean.head(10)
```

Out[33]:

feature	country_code	country_name_wb	year	access_to_clean_fuels_and_technologi
0	AFG	Afghanistan	1960	
1	AFG	Afghanistan	1961	
2	AFG	Afghanistan	1962	
3	AFG	Afghanistan	1963	
4	AFG	Afghanistan	1964	
5	AFG	Afghanistan	1965	
6	AFG	Afghanistan	1966	
7	AFG	Afghanistan	1967	
8	AFG	Afghanistan	1968	
9	AFG	Afghanistan	1969	

10 rows × 74 columns

I.

Originally, I thought that this would a one-to-one merge because it seems that both data frames contain the same information with regard to country_code and year. To test this, I did an outer-merge and tested it with the 'indicator' to see how many are left-only, how many are right-only, and how many are one-to-one. I saw that there were numerous rows that were right only and numerous that were left only.

In [34]:

```
merged_df = pd.merge(wb_clean, vdem_clean, on=['country_code', 'year'], how=
merged_df
```

Out[34]:

	country_code	country_name_wb	year	access_to_clean_fuels_and_technologi
0	AFG	Afghanistan	1960	
1	AFG	Afghanistan	1961	
2	AFG	Afghanistan	1962	
3	AFG	Afghanistan	1963	
4	AFG	Afghanistan	1964	
...	...	...	...	
12549	ZZB	NaN	2017	
12550	ZZB	NaN	2018	
12551	ZZB	NaN	2019	
12552	ZZB	NaN	2020	
12553	ZZB	NaN	2021	

12554 rows × 77 columns

In [35]:
```
print(merged_df['_merge'].value_counts())
```

```
_merge
both          9976
left_only     2183
right_only     395
Name: count, dtype: int64
```

In [36]:
```
left_only = merged_df.query("_merge == 'left_only'")
left_only_years = left_only.groupby(['country_code', 'country_name_wb'])['ye
left_only_years.iloc[150:200]
```

Out[36]:

country_code	country_name_wb	min	max
SEN	Senegal	2022	2022
SGP	Singapore	2022	2022
SLB	Solomon Islands	2022	2022
SLE	Sierra Leone	2022	2022
SLV	El Salvador	2022	2022
SMR	San Marino	1960	2022
SOM	Somalia	2022	2022
SRB	Serbia	2022	2022
SSD	South Sudan	1960	2022
STP	Sao Tome and Principe	2022	2022
SUR	Suriname	2022	2022
SVK	Slovak Republic	1960	2022
SVN	Slovenia	1960	2022
SWE	Sweden	2022	2022
SWZ	Eswatini	2022	2022
SYC	Seychelles	2022	2022
SYR	Syrian Arab Republic	2022	2022
TCD	Chad	2022	2022
TGO	Togo	2022	2022
THA	Thailand	2022	2022
TJK	Tajikistan	1960	2022
TKM	Turkmenistan	1960	2022
TLS	Timor-Leste	2022	2022
TON	Tonga	1960	2022
TTO	Trinidad and Tobago	2022	2022
TUN	Tunisia	2022	2022
TUR	Turkiye	2022	2022
TUV	Tuvalu	1960	2022
TZA	Tanzania	2022	2022
UGA	Uganda	2022	2022
UKR	Ukraine	1960	2022

		min	max
country_code	**country_name_wb**		
URY	**Uruguay**	2022	2022
USA	**United States**	2022	2022
UZB	**Uzbekistan**	1960	2022
VCT	**St. Vincent and the Grenadines**	1960	2022
VEN	**Venezuela, RB**	2022	2022
VNM	**Vietnam**	2022	2022
VUT	**Vanuatu**	2022	2022
WSM	**Samoa**	1960	2022
YEM	**Yemen, Rep.**	2022	2022
ZAF	**South Africa**	2022	2022
ZMB	**Zambia**	2022	2022
ZWE	**Zimbabwe**	2022	2022

In [37]:
```
left_only['country_name_wb'].unique()
```

Out[37]:
```
array(['Afghanistan', 'Angola', 'Albania', 'Andorra',
       'United Arab Emirates', 'Argentina', 'Armenia',
       'Antigua and Barbuda', 'Australia', 'Austria', 'Azerbaijan',
       'Burundi', 'Belgium', 'Benin', 'Burkina Faso', 'Bangladesh',
       'Bulgaria', 'Bahrain', 'Bahamas, The', 'Bosnia and Herzegovina',
       'Belarus', 'Belize', 'Bolivia', 'Brazil', 'Barbados',
       'Brunei Darussalam', 'Bhutan', 'Botswana',
       'Central African Republic', 'Canada', 'Switzerland', 'Chile',
       'China', "Cote d'Ivoire", 'Cameroon', 'Congo, Dem. Rep.',
       'Congo, Rep.', 'Colombia', 'Comoros', 'Cabo Verde', 'Costa Rica',
       'Cuba', 'Cyprus', 'Czechia', 'Germany', 'Djibouti', 'Dominica',
       'Denmark', 'Dominican Republic', 'Algeria', 'Ecuador',
       'Egypt, Arab Rep.', 'Eritrea', 'Spain', 'Estonia', 'Ethiopia',
       'Finland', 'Fiji', 'France', 'Micronesia, Fed. Sts.', 'Gabon',
       'United Kingdom', 'Georgia', 'Ghana', 'Guinea', 'Gambia, The',
       'Guinea-Bissau', 'Equatorial Guinea', 'Greece', 'Grenada',
       'Guatemala', 'Guyana', 'Honduras', 'Croatia', 'Haiti', 'Hungary',
       'Indonesia', 'India', 'Ireland', 'Iran, Islamic Rep.', 'Iraq',
       'Iceland', 'Israel', 'Italy', 'Jamaica', 'Jordan', 'Japan',
       'Kazakhstan', 'Kenya', 'Kyrgyz Republic', 'Cambodia', 'Kiribati',
       'St. Kitts and Nevis', 'Korea, Rep.', 'Kuwait', 'Lao PDR',
       'Lebanon', 'Liberia', 'Libya', 'St. Lucia', 'Liechtenstein',
       'Sri Lanka', 'Lesotho', 'Lithuania', 'Luxembourg', 'Latvia',
       'Morocco', 'Monaco', 'Moldova', 'Madagascar', 'Maldives', 'Mexico',
       'Marshall Islands', 'North Macedonia', 'Mali', 'Malta', 'Myanmar',
       'Montenegro', 'Mongolia', 'Mozambique', 'Mauritania', 'Mauritius',
       'Malawi', 'Malaysia', 'Namibia', 'Niger', 'Nigeria', 'Nicaragua',
       'Netherlands', 'Norway', 'Nepal', 'Nauru', 'New Zealand', 'Oman',
       'Pakistan', 'Panama', 'Peru', 'Philippines', 'Palau',
       'Papua New Guinea', 'Poland', "Korea, Dem. People's Rep.",
       'Portugal', 'Paraguay', 'Qatar', 'Romania', 'Russian Federation',
       'Rwanda', 'Saudi Arabia', 'Sudan', 'Senegal', 'Singapore',
       'Solomon Islands', 'Sierra Leone', 'El Salvador', 'San Marino',
       'Somalia', 'Serbia', 'South Sudan', 'Sao Tome and Principe',
       'Suriname', 'Slovak Republic', 'Slovenia', 'Sweden', 'Eswatini',
       'Seychelles', 'Syrian Arab Republic', 'Chad', 'Togo', 'Thailand',
       'Tajikistan', 'Turkmenistan', 'Timor-Leste', 'Tonga',
       'Trinidad and Tobago', 'Tunisia', 'Turkiye', 'Tuvalu', 'Tanzania',
       'Uganda', 'Ukraine', 'Uruguay', 'United States', 'Uzbekistan',
       'St. Vincent and the Grenadines', 'Venezuela, RB', 'Vietnam',
       'Vanuatu', 'Samoa', 'Yemen, Rep.', 'South Africa', 'Zambia',
       'Zimbabwe'], dtype=object)
```

In [38]:
```
right_only = merged_df.query("_merge == 'right_only'")
right_only_years = right_only.groupby(['country_code', 'country_name_vdem'])
right_only_years
```

Out[38]:

country_code	country_name_vdem	min	max
DDR	German Democratic Republic	1960	1990
HKG	Hong Kong	1960	2021
PSE	Palestine/West Bank	1967	2021
PSG	Palestine/Gaza	1960	2021
SML	Somaliland	1991	2021
TWN	Taiwan	1960	2021
VDR	Republic of Vietnam	1960	1975
XKX	Kosovo	1999	2021
YMD	South Yemen	1960	1990
ZZB	Zanzibar	1960	2021

K.

- There are some countries that have different spellings between data sets. For example, South Yemen in Vdem and Republic of Yemen could refer to the same country, although there are political reasons as to why this may not be true.
- Many of the countries present in the vdem data set that are not present in the wb data set are countries that no longer exist, such as East Germany (German Democratic Republic) or the Republic of Vietnam. In addition, the vdem dataset recognizes Hong Kong, Taiwan, Somolialand while the wb data set does not.
- It appears that the wb_clean data set has more up-to-date statistics than the vdem_clean data set (going up to 2022 as opposed to 2021).

In [39]:

```
timeseries = pd.merge(wb_clean, vdem_clean, on=['country_code', 'year'], how
timeseries
```

Out[39]:

	country_code	country_name_wb	year	access_to_clean_fuels_and_technologie
0	AFG	Afghanistan	1960	
1	AFG	Afghanistan	1961	
2	AFG	Afghanistan	1962	
3	AFG	Afghanistan	1963	
4	AFG	Afghanistan	1964	
...	...	...	...	
9971	ZWE	Zimbabwe	2017	
9972	ZWE	Zimbabwe	2018	
9973	ZWE	Zimbabwe	2019	
9974	ZWE	Zimbabwe	2020	
9975	ZWE	Zimbabwe	2021	

9976 rows × 76 columns

Part 3: 1NF, 2NF, 3NF

In [40]:

```
country = country_clean
country.head(10)
```

Out[40]:

	country_code	country_name_wb	country_longname	currency_unit	region	inc
0	AFG	Afghanistan	Islamic State of Afghanistan	Afghan afghani	South Asia	
1	AGO	Angola	People's Republic of Angola	Angolan kwanza	Sub-Saharan Africa	L
2	ALB	Albania	Republic of Albania	Albanian lek	Europe & Central Asia	U
3	AND	Andorra	Principality of Andorra	Euro	Europe & Central Asia	
5	ARE	United Arab Emirates	United Arab Emirates	U.A.E. dirham	Middle East & North Africa	
6	ARG	Argentina	Argentine Republic	Argentine peso	Latin America & Caribbean	U
7	ARM	Armenia	Republic of Armenia	Armenian dram	Europe & Central Asia	U
8	ATG	Antigua and Barbuda	Antigua and Barbuda	East Caribbean dollar	Latin America & Caribbean	
9	AUS	Australia	Commonwealth of Australia	Australian dollar	East Asia & Pacific	
10	AUT	Austria	Republic of Austria	Euro	Europe & Central Asia	

In [41]:

```
world = whole_world_data_clean
world.tail(10)
```

Out[41]:

feature	year	world_access_to_clean_fuels_and_technologies_for_cooking	world_acce
11897	2013	61.095607	
11898	2014	62.372600	
11899	2015	63.662869	
11900	2016	65.005668	
11901	2017	66.321819	
11902	2018	67.696517	
11903	2019	68.921601	
11904	2020	70.184805	
11905	2021	71.331487	
11906	2022	NaN	

10 rows × 72 columns

In [42]:

```
world.tail(20)
```

Out[42]:

feature	year	world_access_to_clean_fuels_and_technologies_for_cooking	world_acce
11887	2003	51.040051	
11888	2004	51.769412	
11889	2005	52.495195	
11890	2006	53.344280	
11891	2007	54.374873	
11892	2008	55.310321	
11893	2009	56.413537	
11894	2010	57.446700	
11895	2011	58.688718	
11896	2012	59.843821	
11897	2013	61.095607	
11898	2014	62.372600	
11899	2015	63.662869	
11900	2016	65.005668	
11901	2017	66.321819	
11902	2018	67.696517	
11903	2019	68.921601	
11904	2020	70.184805	
11905	2021	71.331487	
11906	2022	NaN	

20 rows × 72 columns

In [43]:

```
timeseries.head(10)
```

Out[43]:

	country_code	country_name_wb	year	access_to_clean_fuels_and_technologies_fo
0	AFG	Afghanistan	1960	
1	AFG	Afghanistan	1961	
2	AFG	Afghanistan	1962	
3	AFG	Afghanistan	1963	
4	AFG	Afghanistan	1964	
5	AFG	Afghanistan	1965	
6	AFG	Afghanistan	1966	
7	AFG	Afghanistan	1967	
8	AFG	Afghanistan	1968	
9	AFG	Afghanistan	1969	

10 rows × 76 columns

Country Dataframe This data frame appears to be in the 1NF because each column has atomic values, there are no repeating groups, and each column has a unique name. Since 'country_code' is a primary key the country data frame also appears to be in 2NF since all other columns (such as 'region') are functionally dependent on the 'country_code'. Finally, this data frame appears to be in 3NF because there does not appear to be any transitive dependency.

Timeseries Dataframe This data frame appears to be in the 1NF because each column has atomic values, there are no repeating groups, and each column has a unique name. There are two primary keys in this data frame, that being 'country_code' and "year". For similar reasons above, this data frame appears to be in 2N and 3N. However, in order for all three dataframes to be in the 3N, then we do not need the country_name_wb nor the country_name_vdem column as this is repeated in the country dataframe.

World Dataframe The world data frame is in 1NF because each column has atomic values, there are no repeating groups, and each column has a unique name. The primary key is the year and the other columns are functionally dependent on the 'year'. Hence, it is in 2NF. Finally, there does not appear to be any transitive dependency.

In [44]:
```
# Dropping the 'country_names_wb' column from Timeseries dataframe.
timeseries = timeseries.drop(['country_name_wb', 'country_name_vdem'], axis=
```

In [45]:
```
timeseries.head(2)
```

Out[45]:

	country_code	year	access_to_clean_fuels_and_technologies_for_cooking	access_t
0	AFG	1960	NaN	
1	AFG	1961	NaN	

2 rows × 74 columns

Task Four

In [46]:
```
lab3network = psycopg2.connect(
    host = 'postgres',
    user = 'postgres',
    password = POSTGRES_PASSWORD,
    port = 5432
)
lab3network.autocommit = True
```

In [47]:
```
cursor = lab3network.cursor()
```

In [48]:
```
try:
    cursor.execute('CREATE DATABASE cardib')
except:
    cursor.execute ('DROP DATABASE cardib')
    cursor.execute ( 'CREATE DATABASE cardib')
```

In [60]:
```
engine = create_engine ('postgresql+psycopg2://{user}:{password}@{host}:{por
user = 'postgres',
password = POSTGRES_PASSWORD,
host = 'postgres',
port = 5432,
db = 'cardib'
))
```

In [50]:
```
print(country.shape[0])
country.to_sql('country', con=engine, index=False, chunksize=1000,
               if_exists = 'replace')
```

193

Out[50]: 193

In [51]:
```
print(timeseries.shape[0])
timeseries.to_sql('timeseries', con=engine, index=False, chunksize=100,
               if_exists = 'replace')
```

9976

Out[51]: 9976

In [52]:
```
print(world.shape[0])
world.to_sql('world', con=engine, index=False, chunksize=1000,
```

```
                if_exists = 'replace')
```

63

Out[52]: 63

In [53]:
```
timeseries.head(10)
```

Out[53]:

	country_code	year	access_to_clean_fuels_and_technologies_for_cooking	access_t
0	AFG	1960	NaN	
1	AFG	1961	NaN	
2	AFG	1962	NaN	
3	AFG	1963	NaN	
4	AFG	1964	NaN	
5	AFG	1965	NaN	
6	AFG	1966	NaN	
7	AFG	1967	NaN	
8	AFG	1968	NaN	
9	AFG	1969	NaN	

10 rows × 74 columns

Task Five

see cardibdb.dbml

https://dbdocs.io/elizabethmillerwc/Cardibi

6.

In [63]:
```
#a. what countries had the highest quality democracies in the year 2021?
myquery = '''
SELECT c.country_name_wb AS country, ts.democracy
FROM timeseries ts
INNER JOIN country c
    ON ts.country_code = c.country_code
WHERE year = 2021
ORDER BY ts.democracy DESC nulls last
'''

pd.read_sql_query(myquery, con=engine)
```

Out[63]:

	country	democracy
0	Denmark	0.915
1	Sweden	0.903
2	Norway	0.901
3	Costa Rica	0.898
4	Switzerland	0.898
...	...	...
167	Qatar	0.090
168	Korea, Dem. People's Rep.	0.086
169	China	0.077
170	Eritrea	0.072
171	Saudi Arabia	0.016

172 rows × 2 columns

In [64]:

```
#b. How does the life expectancy at birth for Chile compare to the glbal ave
myquery = '''
SELECT ts.year, c.country_code, ts.life_expectancy_at_birth, w.world_life_ex
FROM timeseries ts
INNER JOIN country c
    ON ts.country_code = c.country_code
INNER JOIN world w
    ON ts.year = w.year
WHERE c.country_code = 'CHL'
'''
pd.read_sql_query(myquery, con=engine)
```

Out[64]:

	year	country_code	life_expectancy_at_birth	world_life_expectancy_at_birth
0	1960	CHL	57.015	50.894180
1	1961	CHL	57.537	52.846336
2	1962	CHL	57.771	55.208684
3	1963	CHL	57.150	55.542341
4	1964	CHL	58.738	56.034875
...	...	...	...	...
57	2017	CHL	80.350	72.542776
58	2018	CHL	80.133	72.784090
59	2019	CHL	80.326	72.979716
60	2020	CHL	79.377	72.243822
61	2021	CHL	NaN	NaN

62 rows × 4 columns

In [65]:

```
#c. what regions of the world generated the most carbon dioxide emissions ir
myquery = '''
SELECT c.region,
    SUM(co2_emissions) AS co2_emissions
FROM timeseries ts
INNER JOIN country c
    ON ts.country_code = c.country_code
WHERE year = 2019
GROUP BY c.region
ORDER BY co2_emissions DESC
'''

pd.read_sql_query(myquery, con=engine)
```

Out[65]:

	region	co2_emissions
0	Europe & Central Asia	273.807274
1	Middle East & North Africa	176.008097
2	East Asia & Pacific	94.256987
3	Latin America & Caribbean	68.692584
4	Sub-Saharan Africa	43.318399
5	North America	29.726128
6	South Asia	10.820011

In [69]:

```
#d. What countries expereinced the greatest increases in democratic quality
myquery = '''
SELECT c.country_name_wb AS country_name, d.democracy_1960, d.democracy_2021
```

```
FROM (SELECT y2.country_code, y1.democracy_1960, y2.democracy_2021
      FROM (SELECT ts.country_code, ts.democracy AS democracy_2021
            FROM timeseries ts
            WHERE year = 2021) y2
      INNER JOIN (
        SELECT ts.country_code, ts.democracy AS democracy_1960
        FROM timeseries ts
        WHERE year = 1960) y1
        ON y2.country_code = y1.country_code) d
INNER JOIN country c
    ON d.country_code = c.country_code
ORDER BY democracy_diff DESC nulls last
'''
pd.read_sql_query(myquery, con=engine)
```

Out[69]:

	country_name	democracy_1960	democracy_2021	democracy_diff
0	Spain	0.070	0.854	0.784
1	Portugal	0.128	0.888	0.760
2	Cabo Verde	0.023	0.773	0.750
3	Vanuatu	0.080	0.771	0.691
4	Timor-Leste	0.018	0.680	0.662
...	...	...	...	...
143	Lao PDR	0.276	0.135	-0.141
144	Somalia	0.373	0.169	-0.204
145	India	0.669	0.420	-0.249
146	Myanmar	0.421	0.107	-0.314
147	Venezuela, RB	0.648	0.218	-0.430

148 rows × 4 columns

In [70]:

```
#e. By county of countries, what is the most commonly used currency in the w
myquery = '''
SELECT c.currency_unit,
    COUNT(*) AS num_of_countries
FROM country c
GROUP BY c.currency_unit
ORDER BY num_of_countries DESC
LIMIT 10
'''

pd.read_sql_query(myquery, con=engine)
```

Out [70]:

	currency_unit	num_of_countries
0	Euro	24
1	West African CFA franc	8
2	U.S. dollar	7
3	Central African CFA franc	6
4	East Caribbean dollar	6
5	Australian dollar	4
6	Swiss franc	2
7	Sierra Leonean leone	1
8	New Zambian kwacha	1
9	Lao kip	1

In [75]:

```
#f. How does the average GINI index compare across income groups in 2019?
myquery = '''
SELECT
    c.income_group,
    AVG(t.gini_index) as average_gini_index
FROM
    timeseries t
JOIN
    country c ON t.country_code = c.country_code
WHERE
    t.year = 2019
GROUP BY
    c.income_group
ORDER BY
    average_gini_index DESC;
'''
pd.read_sql_query(myquery, con=engine)
```

Out [75]:

	income_group	average_gini_index
0	None	NaN
1	Low income	43.900000
2	Upper middle income	38.145833
3	Lower middle income	37.420000
4	High income	31.731250