Problem 0

In [42]:
```
%matplotlib inline
```

In [ ]:
```
import numpy as np
import pandas as pd
import requests
import os
import io
import zipfile
import psycopg
from sqlalchemy import create_engine
import dotenv
dotenv.load_dotenv()
```

Out[ ]: True

In [2]:
```
#World Bank data
url = 'https://databank.worldbank.org/data/download/ESG_CSV.zip'
r = requests.get(url)
z = zipfile.ZipFile(io.BytesIO(r.content))
z.extractall()
#V-Dem data
url = 'https://www.v-dem.net/media/datasets/V-Dem-CY-Core-v15_csv.zip'
r = requests.get(url)
z = zipfile.ZipFile(io.BytesIO(r.content))
z.extractall()
```

In [3]:
```
vdem = pd.read_csv('V-Dem-CY-Core-v15.csv')
countrydata = pd.read_csv('ESGCountry.csv')
wb = pd.read_csv('ESGCSV.csv')
```

Problem 1

In [4]:
```
# keep only needed columns
vdem = vdem[["country_text_id", "country_name", "year", "v2x_polyarchy"]]

# filter years 1960–2023
vdem = vdem.query("1960 <= year <= 2023")


vdem = vdem.rename(columns={
    "country_text_id": "country_code",
    "country_name": "country_name_vdem",
    "v2x_polyarchy": "democracy"
})

# sort by country_code, year
vdem = vdem.sort_values(["country_code", "year"]).reset_index(drop=True)
```

Problem 2

In [5]:
```
countrydata = countrydata[
    ["Country Code", "Table Name", "Long Name", "Currency Unit", "Region", "Income Group"]
]

countrydata = countrydata.rename(columns={
    "Country Code": "country_code",
    "Table Name": "country_name_wb",
    "Long Name": "country_longname",
    "Currency Unit": "currency_unit",
    "Region": "region",
    "Income Group": "income_group"
})
```

In [6]:
```
noncountries = [
    "Arab World",
    "Central Europe and the Baltics",
    "Caribbean small states",
    "East Asia & Pacific (excluding high income)",
    "Early-demographic dividend",
    "Europe & Central Asia (excluding high income)",
    "Europe & Central Asia",
    "European Union",
    "Fragile and conflict affected situations",
    "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",
    "Low income",
    "Lower middle income",
    "Low & middle income",
    "Late-demographic dividend",
    "Middle East & North Africa",
    "Middle income",
    "Middle East & North Africa (excluding high income)",
    "Middle East, North Africa, Afghanistan & Pakistan",
    "North America",
    "OECD members",
    "Other small states",
    "Pre-demographic dividend",
    "Pacific island small states",
    "Post-demographic dividend",
    "Sub-Saharan Africa (excluding high income)",
    "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 (IDA & IBRD)",
    "Sub-Saharan Africa (IDA & IBRD)"
]

countrydata = countrydata.query("country_name_wb not in @noncountries")
```

Problem 3

In [7]:
```
year_cols = [c for c in wb.columns if c.startswith("19") or c.startswith("20")]

keep_cols = ["Country Code", "Country Name", "Indicator Code"] + year_cols
wb = wb[keep_cols]
```

In [8]:
```
wb = wb.rename(columns={
    "Country Code": "country_code",
    "Country Name": "country_name_wb",
    "Indicator Code": "feature"
})
```

In [11]:
```
#remove 'World' from non-countries before filtering
#noncountries.remove("World")

wb = wb.query("country_name_wb not in @noncountries")
```

In [23]:
```
replace_map = {
'AG.LND.FRLS.HA':'tree_cover_loss_hectares',
'EN.CLC.CSTP.ZS':'coastal_protection',
'EN.CLC.HDDY.XD':'heating_degree_days',
'EN.H2O.BDYS.ZS':'proportion_water_good_quality',
'EN.LND.LTMP.DC':'land_surface_temperature',
'ER.H2O.FWST.ZS':'level_of_water_stress',
'SD.ESR.PERF.XQ':'economic_social_rights_score',
'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_parliaments',
'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_survey_mean_consumption',
'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_rate',
'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'}
```

In [24]:
```
wb["feature"] = wb["feature"].replace(replace_map)
```

Problem 4

Part a

In [25]:
```
wb_long = wb.melt(
    id_vars=["country_code", "country_name_wb", "feature"],
    var_name="variable",
    value_name="value"
)
```

Part b

In [26]:
```
#rename 'variable' to 'year'
wb_long = wb_long.rename(columns={"variable": "year"})

#pivot so each feature becomes a column
wb_ts = wb_long.pivot_table(
    index=["country_code", "country_name_wb", "year"],
    columns="feature",
    values="value"
).reset_index()

#remove column index name for cleanliness
wb_ts.columns.name = None

# replace original wb with this time-series version
wb = wb_ts
```

Part c

In [27]:
```
wb["year"] = wb["year"].astype(int)
```

Part d

In [28]:
```
# rows for 'World'
world = wb.query('country_name_wb == "World"').copy()

# drop them from country-level data
wb = wb.query('country_name_wb != "World"').copy()

# drop country code/name from world table
```

```
world = world.drop(columns=["country_code", "country_name_wb"])

# prefix all non-year columns with 'world_'
world = world.rename(columns={
    col: f"world_{col}" for col in world.columns if col != "year"
})
```

In [29]:
```
wb
```

Out[29]:

	country_code	country_name_wb	year	access_to_clean_fuels_and_technologies_for_cooking	access_to_electricity	agricultural_land
0	AFG	Afghanistan	1960	NaN	NaN	NaN
1	AFG	Afghanistan	1961	NaN	NaN	57.878356
2	AFG	Afghanistan	1962	NaN	NaN	57.955016
3	AFG	Afghanistan	1963	NaN	NaN	58.031676
4	AFG	Afghanistan	1964	NaN	NaN	58.116002
...	...	...	...	...	...	...
12987	ZWE	Zimbabwe	2019	30.3	46.7	39.518358
12988	ZWE	Zimbabwe	2020	30.5	52.7	39.754073
12989	ZWE	Zimbabwe	2021	30.5	49.0	39.385906
12990	ZWE	Zimbabwe	2022	30.8	50.1	39.489284
12991	ZWE	Zimbabwe	2023	NaN	62.0	NaN

12928 rows × 74 columns

In [30]:
```
vdem
```

Out[30]:

	country_code	country_name_vdem	year	democracy
0	AFG	Afghanistan	1960	0.082
1	AFG	Afghanistan	1961	0.083
2	AFG	Afghanistan	1962	0.083
3	AFG	Afghanistan	1963	0.086
4	AFG	Afghanistan	1964	0.139
...	...	...	...	...
10724	ZZB	Zanzibar	2019	0.266
10725	ZZB	Zanzibar	2020	0.271
10726	ZZB	Zanzibar	2021	0.285
10727	ZZB	Zanzibar	2022	0.294
10728	ZZB	Zanzibar	2023	0.298

10729 rows × 4 columns

Problem 5

Part a

I would expect a one to one merge because both the 'wb' and 'vdem' dataframes seem to have at most one row per country per year after cleaning.

Part b

In [ ]:
```
merged = wb.merge(
    vdem,
    on=["country_code", "year"],
    how="outer",
    indicator=True,
    validate="one_to_one"
)
```

Part c

In [32]:
```
merged["_merge"].value_counts()
```

Out[32]:
```
_merge
both          10320
left_only      2608
right_only      409
Name: count, dtype: int64
```

In [33]:
```
#present in wb but not vdem
left_only = (
    merged.query('_merge == "left_only"')
          .groupby(["country_code", "country_name_wb"])["year"]
          .agg(["min", "max"])
          .reset_index()
)

#present in vdem but not wb
right_only = (
    merged.query('_merge == "right_only"')
          .groupby(["country_code", "country_name_vdem"])["year"]
          .agg(["min", "max"])
          .reset_index()
)
```

Part d

- No significant name-mismatch cases were found.
- 'wb' dataframe contains mostly small states, territories, or early subnational units within empires or unions (micro-states, Gulf monarchies, pre-independence regions).
- 'vdem' dataframe contains mostly historical or partially recognized states (East Germany, South Vietnam, Kosovo, Somaliland, Taiwan).

These differences arise from the distinct institutional definitions of the countries. The World Bank includes only recognized sovereign members and economic territories. V-Dem includes any political unit for which it can construct democracy quality estimates, even if unrecognized or defunct.

Part e

In [35]:
```
timeseries = wb.merge(
    vdem,
    on=["country_code", "year"],
    how="inner"
)
```

In [36]:
```
timeseries.shape
```

Out[36]:
```
(10320, 76)
```

Problem 6

Part a

My schema here uses three tables, country (one row per country_code), world (one row per year, world-level indicators), and timeseries (one row per (country_code,year) with WB indicators and V-Dem democracy). All three tables satisfy 1NF (atomic values, no repeating groups). They also hold 2NF because each non key attribute depends on the full key of its table. In country the key is 'country_code'. In world the key is 'year'. In timeseries the composite key is ('country_code','year'). To achieve 3NF, I would have to eliminate transitive dependencies by:

- keeping only one country name in the database (store names in country and drop country_name_wb/country_name_vdem from timeseries)
- ensuring the world table truly contains only world rows (filter to country_name_wb == 'World', drop country_code/country_name_wb, and keep year as the sole key).

After these adjustments, all non key attributes would be expected to depend only on their table's key and not on other non-key attributes. Then the design would reach Third Normal Form (3NF).

In [ ]:
```
country = countrydata.copy()

if "country_name_vdem" in timeseries.columns:
    timeseries = timeseries.drop(columns=["country_name_vdem"])
```

In [41]:
```
timeseries
```

Out[41]:

	country_code	country_name_wb	year	access_to_clean_fuels_and_technologies_for_cooking	access_to_electricity	agricultural_land
0	AFG	Afghanistan	1960	NaN	NaN	NaN
1	AFG	Afghanistan	1961	NaN	NaN	57.878356
2	AFG	Afghanistan	1962	NaN	NaN	57.955016
3	AFG	Afghanistan	1963	NaN	NaN	58.031676
4	AFG	Afghanistan	1964	NaN	NaN	58.116002
...	...	...	...	...	...	...
10315	ZWE	Zimbabwe	2019	30.3	46.7	39.518358
10316	ZWE	Zimbabwe	2020	30.5	52.7	39.754073
10317	ZWE	Zimbabwe	2021	30.5	49.0	39.385906
10318	ZWE	Zimbabwe	2022	30.8	50.1	39.489284
10319	ZWE	Zimbabwe	2023	NaN	62.0	NaN

10320 rows × 75 columns

Part b

```
(base) digifort@digifort:~/Documents/Data_Management_F25/Anns_repo_1$ docker rm -f cardib-postgres
docker run --name cardib-postgres \
  -e POSTGRES_PASSWORD=Jayangdehi@2025 \
  -p 5432:5432 \
  -d postgres
cardib-postgres
e90893142807a2d70e934354b12778c7f69b526aec803b7e369f368d46f56ec8
(base) digifort@digifort:~/Documents/Data_Management_F25/Anns_repo_1$
```

In [55]:
```python
from sqlalchemy import create_engine
from sqlalchemy.engine import URL

url = URL.create(
    drivername="postgresql+psycopg",
    username="postgres",
    password="Jayangdehi@2025",   # raw string OK here
    host="127.0.0.1",             # <-- NO :5433 here
    port=5433,                    # <-- port goes here
    database="postgres",
)
engine = create_engine(url, pool_pre_ping=True)

# smoke test
with engine.connect() as conn:
    print(conn.exec_driver_sql("SELECT 1").scalar())
```

1

In [56]:
```python
import psycopg

with psycopg.connect(
    dbname="postgres",
    user="postgres",
    password="Jayangdehi@2025",
    host="127.0.0.1",
    port=5433,
    autocommit=True
) as conn:
    with conn.cursor() as cur:
        cur.execute("DROP DATABASE IF EXISTS cardib;")
        cur.execute("CREATE DATABASE cardib;")

# new engine pointed at cardib
url = url.set(database="cardib")
engine = create_engine(url, pool_pre_ping=True)
```

In [57]:
```python
country.to_sql("country", engine, index=False, if_exists="replace")
world.to_sql("world", engine, index=False, if_exists="replace")
timeseries.to_sql("timeseries", engine, index=False, if_exists="replace")
```

Out[57]: -1

Part c

In [58]:
```python
def map_pandas_dtype_to_dbml_type(dtype) -> str:
    dtype_name = str(dtype)
    if "int" in dtype_name:
        return "int"
    if "float" in dtype_name:
```

```
        return "float"
    if "datetime" in dtype_name:
        return "datetime"
    return "varchar"

def pandas_df_to_dbml(df: pd.DataFrame, table_name: str) -> str:
    dbml_string = f"Table {table_name} {{\n"
    for column_name, column_type in df.dtypes.items():
        dbml_type = map_pandas_dtype_to_dbml_type(column_type)
        dbml_string += f"  {column_name} {dbml_type}\n"
    dbml_string += "}\n"
    return dbml_string
```

In [59]:
```
print(pandas_df_to_dbml(country, "country"))
print(pandas_df_to_dbml(world, "world"))
print(pandas_df_to_dbml(timeseries, "timeseries"))
```

```
Table country {
  country_code varchar
  country_name_wb varchar
  country_longname varchar
  currency_unit varchar
  region varchar
  income_group varchar
}

Table world {
  year int
  world_access_to_clean_fuels_and_technologies_for_cooking float
  world_access_to_electricity float
  world_agricultural_land float
  world_agriculture float
  world_annual_freshwater_withdrawals float
  world_annualized_average_growth_rate_in_per_capita_real_survey_mean_consumption float
  world_cause_of_death float
  world_children_in_employment float
  world_co2_emissions float
  world_coastal_protection float
  world_control_of_corruption float
  world_cooling_degree_days float
  world_economic_social_rights_score float
  world_electricity_production_from_coal_sources float
  world_energy_imports float
  world_energy_intensity_level_of_primary_energy float
  world_energy_use float
  world_fertility_rate float
  world_food_production_index float
  world_forest_area float
  world_fossil_fuel_energy_consumption float
  world_gdp_growth float
  world_ghg_net_emissions float
  world_gini_index float
  world_government_effectiveness float
  world_government_expenditure_on_education float
  world_gross_school_enrollment float
  world_heat_index_35 float
  world_heating_degree_days float
  world_hospital_beds float
  world_income_share_held_by_lowest_20pct float
  world_individuals_using_the_internet float
  world_labor_force_participation_rate float
  world_land_surface_temperature float
  world_level_of_water_stress float
  world_life_expectancy_at_birth float
  world_literacy_rate float
  world_mammal_species float
  world_mean_drought_index float
  world_methane_emissions float
  world_mortality_rate float
  world_natural_resources_depletion float
  world_net_forest_depletion float
  world_net_migration float
  world_nitrous_oxide_emissions float
  world_patent_applications float
  world_people_using_safely_managed_drinking_water_services float
  world_people_using_safely_managed_sanitation_services float
  world_pm2_5_air_pollution float
  world_political_stability_and_absence_of_violence float
  world_population_ages_65_and_above float
  world_population_density float
  world_poverty_headcount_ratio_at_national_poverty_lines float
  world_prevalence_of_overweight float
  world_prevalence_of_undernourishment float
  world_primary_school_enrollment float
  world_proportion_of_seats_held_by_women_in_national_parliaments float
  world_proportion_water_good_quality float
  world_ratio_of_female_to_male_labor_force_participation_rate float
  world_regulatory_quality float
  world_renewable_electricity_output float
  world_renewable_energy_consumption float
  world_research_and_development_expenditure float
  world_rule_of_law float
  world_scientific_and_technical_journal_articles float
  world_strength_of_legal_rights_index float
  world_terrestrial_and_marine_protected_areas float
  world_tree_cover_loss_hectares float
  world_unemployment float
  world_unmet_need_for_contraception float
  world_voice_and_accountability float
}

Table timeseries {
  country_code varchar
  country_name_wb varchar
```

```
  year int
  access_to_clean_fuels_and_technologies_for_cooking float
  access_to_electricity float
  agricultural_land float
  agriculture float
  annual_freshwater_withdrawals float
  annualized_average_growth_rate_in_per_capita_real_survey_mean_consumption float
  cause_of_death float
  children_in_employment float
  co2_emissions float
  coastal_protection float
  control_of_corruption float
  cooling_degree_days float
  economic_social_rights_score float
  electricity_production_from_coal_sources float
  energy_imports float
  energy_intensity_level_of_primary_energy float
  energy_use float
  fertility_rate float
  food_production_index float
  forest_area float
  fossil_fuel_energy_consumption float
  gdp_growth float
  ghg_net_emissions float
  gini_index float
  government_effectiveness float
  government_expenditure_on_education float
  gross_school_enrollment float
  heat_index_35 float
  heating_degree_days float
  hospital_beds float
  income_share_held_by_lowest_20pct float
  individuals_using_the_internet float
  labor_force_participation_rate float
  land_surface_temperature float
  level_of_water_stress float
  life_expectancy_at_birth float
  literacy_rate float
  mammal_species float
  mean_drought_index float
  methane_emissions float
  mortality_rate float
  natural_resources_depletion float
  net_forest_depletion float
  net_migration float
  nitrous_oxide_emissions float
  patent_applications float
  people_using_safely_managed_drinking_water_services float
  people_using_safely_managed_sanitation_services float
  pm2_5_air_pollution float
  political_stability_and_absence_of_violence float
  population_ages_65_and_above float
  population_density float
  poverty_headcount_ratio_at_national_poverty_lines float
  prevalence_of_overweight float
  prevalence_of_undernourishment float
  primary_school_enrollment float
  proportion_of_seats_held_by_women_in_national_parliaments float
  proportion_water_good_quality float
  ratio_of_female_to_male_labor_force_participation_rate float
  regulatory_quality float
  renewable_electricity_output float
  renewable_energy_consumption float
  research_and_development_expenditure float
  rule_of_law float
  scientific_and_technical_journal_articles float
  strength_of_legal_rights_index float
  terrestrial_and_marine_protected_areas float
  tree_cover_loss_hectares float
  unemployment float
  unmet_need_for_contraception float
  voice_and_accountability float
  democracy float
}
```

Published at: https://dbdiagram.io/d/Wb_Vdem-69211d30228c5bbc1afce299

Problem 7

Part a

```sql
SELECT
    c.country_name_wb AS country,
```

```
    t.democracy
FROM timeseries AS t
JOIN country AS c
  ON t.country_code = c.country_code
WHERE t.year = 2023
  AND t.democracy IS NOT NULL
ORDER BY t.democracy DESC;
```

country	democracy
Denmark	0.916
Belgium	0.898
Estonia	0.895
Ireland	0.895
Switzerland	0.887
Norway	0.884
Sweden	0.881
New Zealand	0.878
France	0.876
Luxembourg	0.874

Top ten countries displayed

Part b

```
SELECT
    t.year,
    t.life_expectancy_at_birth AS chile_life_expectancy,
    w.world_life_expectancy_at_birth AS world_life_expectancy
FROM timeseries AS t
JOIN world AS w
  ON t.year = w.year
WHERE t.country_code = 'CHL'
ORDER BY t.year;
```

year	chile_life_expectancy	world_life_expectancy
1960	57.216	50.94188607453528
1961	57.309	52.79723012782893
1962	57.811	55.28607340265339
1963	58.349	55.6522285444853
1964	58.812	56.09669797197644
1965	59.473	55.92644764392398
1966	60.373	56.449007563588246
1967	61.054	56.90375264939781
1968	61.68	57.3352975180805
1969	62.341	57.68007855640185

Top 10 entries shown

Part c

```
SELECT
    c.region,
    SUM(t.co2_emissions) AS co2_emissions
FROM timeseries AS t
JOIN country AS c
  ON t.country_code = c.country_code
WHERE t.year = 2020
GROUP BY c.region
ORDER BY co2_emissions DESC;
```

region	co2_emissions
Europe & Central Asia	251.37633782050614
Middle East, North Africa, Afghanistan & Pakistan	171.49161286691006
East Asia & Pacific	88.66708994315316
Latin America & Caribbean	61.54913705888712
Sub-Saharan Africa	39.64624956671595
North America	26.62192195557938
South Asia	7.8360943023082115

Part d

```
WITH dem_1960 AS (
    SELECT
        country_code,
        democracy AS democracy_1960
    FROM timeseries
```

```
    WHERE year = 1960
),
dem_2023 AS (
    SELECT
        country_code,
        democracy AS democracy_2023
    FROM timeseries
    WHERE year = 2023
)
SELECT
    c.country_name_wb AS country,
    d60.democracy_1960,
    d23.democracy_2023,
    (d23.democracy_2023 - d60.democracy_1960) AS democracy_change
FROM dem_1960 AS d60
JOIN dem_2023 AS d23
  ON d60.country_code = d23.country_code
JOIN country AS c
  ON c.country_code = d60.country_code
ORDER BY democracy_change DESC
LIMIT 10;
```

country	democracy_1960	democracy_2023	democracy_change
Spain	0.07	0.85	0.78
Cabo Verde	0.02	0.757	0.737
Vanuatu	0.08	0.815	0.735
Portugal	0.128	0.851	0.723
Timor-Leste	0.018	0.712	0.694
Czechia	0.195	0.869	0.6739999999999999
São Tomé and Principe	0.028	0.673	0.645
Seychelles	0.128	0.738	0.61
Lesotho	0.089	0.675	0.5860000000000001
Namibia	0.063	0.627	0.5640000000000001

Part e

```
SELECT
    currency_unit,
    COUNT(*) AS num_countries
FROM country
WHERE currency_unit IS NOT NULL
GROUP BY currency_unit
ORDER BY num_countries DESC
LIMIT 1;
```

currency_unit	num_countries
Euro	24

Part f

```
SELECT
    c.income_group,
    AVG(t.gini_index) AS avg_gini_2022
FROM timeseries AS t
JOIN country AS c
  ON t.country_code = c.country_code
WHERE t.year = 2022
  AND t.gini_index IS NOT NULL
GROUP BY c.income_group
ORDER BY avg_gini_2022 DESC;
```

income_group	avg_gini_2022
Upper middle income	38.785
Lower middle income	33.75555555555555
High income	32.67666666666666
Low income	32

Part g

```
WITH republics AS (
    SELECT
        'Republic_or_Democratic' AS group_name,
        AVG(t.democracy) AS avg_democracy_2023
    FROM timeseries AS t
    JOIN country AS c
      ON t.country_code = c.country_code
```

```
    WHERE t.year = 2023
      AND t.democracy IS NOT NULL
      AND (
            c.country_longname ILIKE '%Republic%'
         OR c.country_longname ILIKE '%Democratic%'
      )
)
SELECT * FROM republics

UNION ALL

SELECT
    'Other' AS group_name,
    AVG(t.democracy) AS avg_democracy_2023
FROM timeseries AS t
JOIN country AS c
  ON t.country_code = c.country_code
WHERE t.year = 2023
  AND t.democracy IS NOT NULL
  AND (
        c.country_longname NOT ILIKE '%Republic%'
    AND c.country_longname NOT ILIKE '%Democratic%'
  );
```

```
       group_name       | avg_democracy_2023
------------------------+--------------------
Republic_or_Democratic |           0.489175
Other                  |  0.518923076923077
```