Lab Assignment 3

DS 6600: Data Engineering 1

100 points

Instructions

Please answer the following questions as completely as possible using text, code, and the results of code as needed. Format your answers in a Jupyter notebook. To receive full credit, make sure you address every part of the problem, and make sure your document is formatted in a clean and professional way.

Use of AI is allowed, but restricted. **Please read the syllabus for specific guidelines on use of AI for this lab assignment.**

In this lab, we are going to build the Country Analysis Relational DataBase (which we will call the C.A.R.D.B. or the "Cardi B"):

This lab will walk through the entire data pipeline, starting by loading data from original sources, using `pandas` to wrangle the raw data including merging and reshaping dataframes, organizing the data in third normal form, building a PostgreSQL database, documenting it with an ER diagram, and using SQL to query it. The result is an analytics-ready database of many features from nearly every country in the world since 1960.

This lab is designed to follow the discussions of chapters 6 (Database Design), 7 (SQL), 8 (pandas), and 9 (pandas merges and reshaping) of Surfing the Data Pipeline with Python. I recommend that you have the textbook open while you do this lab.

We will be collecting data from two sources. First, we will use open data from the World Bank's Sovereign Environmental, Social, and Governance (ESG) Data project. The ESG data reports data from every country in the world over the time frame from 1960-2023 on a wide variety of topics including education, health, and economic factors within the countries. Second, we will use data on the quality and democratic character of countries' governments as reported by the Varieties of Democracy (V-Dem) project at the University of Notre Dame. By using both data sources, we can conduct analyses to see whether democratic openness leads to better societal outcomes for countries. We can also write queries to capture a wide range of information on countries' political parties, tax systems, and banking industries, for example. Or as Cardi B would say, "You in the club just to party, I'm there, I get paid a fee. I be in and out them banks so much, I know they're tired of me."

Problem 0

Import the following packages:

In [21]:
```
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[21]:
```
True
```

To obtain the data, run the following code cell one time. The code downloads two zipped directories for the World Bank and V-Dem datasets and unzips these files. After running this code, you should have files named "ESGCSV.csv", "ESGCountry.csv", and "V-Dem-CY-Core-v15.csv" in your working directory (and a few additional ESG data files). After confirming that these files are present, comment out or delete this code cell so that you don't download the data again unnecessarily:

In [22]:
```
#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()
```

You will only need three of the files you've downloaded:

In [23]:
```
vdem = pd.read_csv('V-Dem-CY-Core-v15.csv')
countrydata = pd.read_csv('ESGCountry.csv')
wb = pd.read_csv('ESGCSV.csv')
```

Problem 1

First, let's focus on the `vdem` data. Use `pandas` methods to perform the following tasks:

- Keep only the 'country_text_id', 'country_name','year', and 'v2x_polyarchy' columns.
- Use the `.query()` method to keep only the rows in which year is greater than or equal to 1960 and less than or equal to 2023.
- Rename 'country_text_id' to 'country_code', 'country_name' to 'country_name_vdem', and 'v2x_polyarchy' to 'democracy'.
- Sort the rows by 'country_code' and 'year' in ascending order for both columns. [4 points]

Problem 2

Next let's focus on the `countrydata` data. Use `pandas` methods to perform the following tasks:

- Keep only the 'Country Code', 'Table Name', 'Long Name', 'Currency Unit', 'Region', and 'Income Group' columns.
- Rename 'Country Code' to 'country_code', 'Table Name' to 'country_name_wb', 'Long Name' to 'country_longname', 'Currency Unit' to 'currency_unit', 'Region' to 'region', and 'Income Group' to 'income_group'.
- The countries in this dataset include various grouping of countries such as "Europe

& Central Asia". We want to remove these observations from the data. Use the .query() method to remove the rows in which 'country_name_wb' is equal to one of these non-countries. (Hint: within the .query() method, you can reference an external Python variable such as `noncountries` by placing an @ sign in front of the variable name.) Please use this list of non-countries: [4 points]

In [ ]:
```
noncountries = ["Arab World", "Central Europe and the Baltics","Caribbean sm
                "East Asia & Pacific (excluding high income)","Early-demogra
                "Europe & Central Asia (excluding high income)","Europe & Ce
                "European Union","Fragile and conflict affected situations",
                "Heavily indebted poor countries (HIPC)","IBRD only","IDA &
                "IDA total","IDA blend","IDA only",
                "Latin America & Caribbean (excluding high income)","Latin A
                "Low income","Lower middle income","Low & middle income",
                "Late-demographic dividend","Middle East & North Africa","Mi
                "Middle East & North Africa (excluding high income)",
                "Middle East, North Africa, Afghanistan & Pakistan",
                "North America","OECD members",
                "Other small states","Pre-demographic dividend","Pacific isl
                "Post-demographic dividend","Sub-Saharan Africa (excluding h
                "Small states","East Asia & Pacific (IDA & IBRD)","Europe &
                "Latin America & Caribbean (IDA & IBRD)","Middle East & Nort
                "South Asia (IDA & IBRD)","Sub-Saharan Africa (IDA & IBRD)",
```

Problem 3

Next focus on the `wb` data. Use `pandas` methods to perform the following tasks:

- Keep only the columns named 'Country Code', 'Country Name', and 'Indicator Code', or begin with '19' or '20'. (Don't type in all the years individually. Instead, use code that finds all columns that begin '19' or '20'.)
- Rename 'Country Code' to'country_code', 'Country Name' to 'country_name_wb', and 'Indicator Code' to 'feature'.
- Type `noncountries.remove('World')` to remove 'World' from the `noncountries` list. (We want to keep the total world data for now). Then use the `.query()` method to remove the rows in which 'country_name_wb' is equal to one of the other entries in the `noncountries` list.
- The features in this dataset are given strange and incomprehensible codes such as 'EG.CFT.ACCS.ZS'. Use the `replace_map` dictionary, defined below, to recode all of these values with more descriptive names for each feature. [4 points]

In [27]:
```
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_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_survey
'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_rat
'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'}
```

Problem 4

The `wb` dataset is strangely organized. The features are stored in the rows, when typically we would want these features to be columns. Also, years are stored in columns, when typically we would want years to be represented by different rows. We can repair this structure by reshaping the data.

Part a

First, reshape the data to turn the columns that refer to years into rows. [6 points]

Part b

Then rename `variable` to `year`, and reshape the data again by turning the rows that refer to features into columns. [6 points]

Part c

After these reshapes, the year column in the `wb` data frame is stored as a string. Convert this column to an integer data type. [2 points]

Part d

Create a new dataframe containing just the rows from the `wb` dataset that refer to the whole world. Then remove the "World" rows from the `wb` dataset. Drop the 'country_code' and 'country_name_wb' from the world dataset. Finally, with the exception of the year column, rename all columns in the world dataset by adding "world_" in front of the column name. [2 points]

Problem 5

Next we will merge the `wb` data frame with the `vdem` data frame, matching on the 'country_code' and 'year' columns.

Part a

First, write a sentence stating whether you expect this merge to be one-to-one, many-to-one, one-to-many, or many-to-many, and describe your rationale. [2 points]

Part b

Next, merge the two datasets together in a way that checks whether your expectation is met, and also allows you to see the rows that failed to match. [6 points]

Part c

After this merge, use the `.value_counts()` method to see the total number of observations that were found in both datasets, the number found only in the left dataset, and the number found only in the right dataset. (If you entered the `wb` data frame into the merge function first, then "left_only" refers to the rows found in the World Bank but not V-Dem, and "right_only" refers to the rows found in V-Dem but not the World Bank.) There should be more than 10,000 rows that matched, but more than 2000 that failed to match because they were found in `wb` but not `vdem` (left_only), and more than 400 that failed to match because they were found in `vdem` but not `wb` (right_only).

Then conduct two data aggregations to help us investigate why these observations did not match:

- First use `.query()` to keep only the observations that were present in `wb` but not `vdem`. Use `.groupby()` to aggregate the data by both 'country_code' and 'country_name_wb'. Then save the minimum and maximum values of 'year' for each country.
- Then use `.query()` to keep only the observations that were present in `vdem` data but not `wb`. Use `.groupby()` to aggregate the data by both 'country_code' and 'country_name_vdem'. Then save the minimum and maximum values of 'year' for each country. [6 points]

Part d

Here's where a deep understanding of the data becomes very important. There are two reasons why an observation may fail to match in a merge. One reason is a difference in spelling. Suppose that South Korea (which is also known as the Republic of Korea) is coded as SKO in the World Bank data and ROK in V-Dem. In this case, we should recode one or the other of SKO and ROK so that they match, otherwise we will lose the data on

South Korea. But the second reason why observations might fail to match is due to differences in coverage in the data collection strategy: it is possible that a country wasn't included in one data's coverage, or that certain years for that country were not included. For differences in coverage, there's no way to manipulate the data to match, so we are out of luck and we have to either delete these observations or proceed with missing data from one of the data sources.

Take a close look at the two data aggregation tables you generated in part (c), and answer the following questions:

- Do you see any countries that are present in both the unmatched World Bank rows and the unmatched V-Dem rows, but with different spellings?
- Do some more digging on Wikipedia and other sources on the Internet. What do you think is the primary reason why some countries are present in the World Bank data but not V-Dem? (You don't need to describe the reasoning for every country. Just dig until you see a general pattern and describe it here.)
- Do some digging on Wikipedia and other sources on the Internet. What do you think is the primary reason why some countries are present in the V-Dem data but not the World Bank? (You don't need to describe the reasoning for every country. Just dig until you see a general pattern and describe it here.)

[4 points]

Part e

Once you are convinced that all of the unmatched observations are due to differences in the coverage of the data collection strategies of the World Bank and V-Dem, repeat the merge, dropping all unmatched observations. This time there is no need to validate the type of merge, and no need to define a variable to indicate matching. [2 points]

Problem 6

After working through all of the data wrangling steps in the previous problems, you now have the following data frames:

- The World Bank's country level indicators (`countrydata`)
- The World Bank's indicators for the whole world through the 1960-2023 timeframe
- The merged data frame containing both the World Bank's and V-Dem's country-time-series data

Part a

Do these three tables together comprise a third normal form database? Write a paragraph that describes your reasoning, and be clear about whether the criteria or 1st, 2nd, and 3rd normal forms have been achieved. If the data are not in 3rd normal form, take steps to put the data in 3rd normal form. (Hint: it's not important to keep the country names from both the World Bank and V-Dem. If one of these names is stored in one of the other tables, that works too.) [6 points]

Part b

Use Docker to run a PostgreSQL server on your local computer on port 5432. Connect to this server using the `psycopg` package and create a new database called 'cardib'. Then use sqlalchemy to create an engine that connects to this database, and upload the country, world, and merged World Bank and V-Dem (call this one the "timeseries" table in the database) tables to this database. [6 points]

Part c

Use dbdiagram.io and the python functions on the "Using dbdiagram.io and dbdocs.io to Document Your Database" page under Modules on Canvas to create an ER diagram for this database. Copy the text of your .dbml file and paste it in a "raw" type cell below. Either publish your diagram to dbdocs.io or use the sharing function on dbdiagram.io to get a stable URL for the diagram, and paste this URL in a markdown cell.

A few points to keep in mind:

- In the timeseries table, country_code is varchar, year is int, and every feature is float.
- Pay close attention to whether the tables have a one-to-one, many-to-one, or one-to-many relationship with each other, and make sure the ER diagram properly represents that relationship.
- It would be good to write a sentence or two describing the sources of the data as a note for the database.

As writing all the columns in a DBML file can be tedious, here is a helper function. The input is a dataframe, and the output (when passed to the `print()` function) can be copied-and-pasted to the DBML file. [6 points]

In [42]:
```
def pandas_df_to_dbml(df: pd.DataFrame, table_name: str) -> str:
    """
    Converts a pandas DataFrame to a DBML string.

    Args:
        df: The pandas DataFrame to convert.
        table_name: The name of the table in the DBML schema.
```

```
    Returns:
        A DBML string representing the DataFrame schema.
    """

    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

def map_pandas_dtype_to_dbml_type(dtype) -> str:
    """Maps a pandas dtype to a DBML type."""
    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"
```

Problem 7

Write SQL queries for the cardib database that perform the following tasks, and display the output:

Part a

What countries had the highest quality democracies in the year 2023? Join the timeseries and country tables, select the country_name_wb and democracy attributes, rename county_name_wb to country, filter the rows to only those for the year 2023 where the democracy score is not missing, and sort the rows in descending order by the value of democracy. [4 points]

Part b

How does the life expectancy at birth for Chile compare to the global average life expectancy at birth over the 1960-2023 time span? Join the timeseries and world tables, select year from the timeseries table and the life expectancy attributes from both the timeseries and world tables, filter the rows to just those where the country code is CHL, and sort the rows by year. [4 points]

Part c

What regions of the world generated the most carbon dioxide emissions in 2020? Join the timeseries and country tables, filter the rows to only the ones that refer to 2020, and aggregate the data by region. Keep the region column from the country table and the sum of the co2_emissions attribute from the timeseries table. Rename the sum to co2_emissions. Then sort the rows in descending order by co2_emissions. [4 points]

Part d

What countries experienced the greatest increases in democratic quality over the 1960-2023 time span? One approach is to use subqueries to create two new tables on the fly. One table contains country_code and democracy from the timeseries table in which only the rows from 1960 are present. The second table contains country_code and democracy from the timeseries table in which only the rows from 2023 are present. Join these two tables together, then join the result with the country table. Select the country name from county, democracy from the 1960 table, democracy from the 2023 table, and the difference between the two democracy scores. Give each column a more intuitive name. Then sort the rows in descending order by the democracy difference and display the first 10 rows. [4 points]

For the next three questions, write a query that answers the given question without additional guidance.

Part e

By count of countries, what is the most commonly used currency in the world? [6 points]

Part f

The GINI index measures the amount of economic inequality in a country. The higher the index, the greater the economic disparity between rich and poor. How does the average GINI index compare across income groups (see the country table) in 2022? [6 points]

Part g

Some countries include the word 'Republic' or 'Democratic' in their official names. Other countries do not. Which of these two groups of countries had a higher average level of democratic quality in 2023? (It's OK if you use two queries to provide this answer.) [6 points]

In [ ]: