```
# ------------------------------------------------------------
# GDP_Deflator2.py  (Annual GDP, Reference Year 2020 + Chart)
# ------------------------------------------------------------
# Downloads annual Nominal (CP_MEUR) and Real (CLV20_MEUR) GDP
# for Greece, Germany, Italy, and Spain from Eurostat.
# Computes Deflator = Nominal / Real, saves to Excel,
# and embeds a chart of the deflators.
# ------------------------------------------------------------

import requests
import pandas as pd
from openpyxl import load_workbook
from openpyxl.chart import LineChart, Reference

# --- 1. Setup ---
countries = {"EL": "Greece", "DE": "Germany", "IT": "Italy", "ES": "Spain"}
url = "https://ec.europa.eu/eurostat/api/dissemination/statistics/1.0/data/nama_10_gdp"
excel_file = "GDP_Deflator2.xlsx"

def get_gdp(geo_code, country_name, unit):
    """Fetch annual GDP for a given country/unit from Eurostat."""
    params = {
        "na_item": "B1GQ",   # GDP at market prices
        "unit": unit,        # CP_MEUR (nominal) or CLV20_MEUR (real, ref 2020)
        "geo": geo_code
```

```
}
r = requests.get(url, params=params)
r.raise_for_status()
data = r.json()
if "value" not in data or not data["value"]:
    print(f"No data found for {country_name} ({unit}).")
    return pd.DataFrame()

years = list(data["dimension"]["time"]["category"]["label"].keys())
years.sort()
val_dict = data["value"]

records = []
for y in years:
    idx = data["dimension"]["time"]["category"]["index"][y]
    val = val_dict.get(str(idx))
    if val is not None:
        records.append((int(y), val))

df = pd.DataFrame(records, columns=["Year", f"{country_name} ({unit})"])
df.set_index("Year", inplace=True)
df[f"{country_name} ({unit})"] = df[f"{country_name} ({unit})"].astype(float)
print(f"Downloaded {country_name} ({unit}) successfully.")
return df
```

```
# --- 2. Download Nominal and Real GDP ---
print("Downloading annual GDP data (reference year 2020)...")

nominal = [get_gdp(c, n, "CP_MEUR") for c, n in countries.items()]
real = [get_gdp(c, n, "CLV20_MEUR") for c, n in countries.items()]

# Merge across countries
df_nominal = pd.concat(nominal, axis=1)
df_real = pd.concat(real, axis=1)

# --- 3. Combine and Compute Deflator ---
df_combined = pd.concat([df_nominal, df_real], axis=1)

print("Calculating GDP Deflator = Nominal / Real ...")
df_deflator = pd.DataFrame(index=df_combined.index)
for n in countries.values():
    nom_col = f"{n} (CP_MEUR)"
    real_col = f"{n} (CLV20_MEUR)"
    if nom_col in df_combined.columns and real_col in df_combined.columns:
        df_deflator[f"{n} Deflator"] = df_combined[nom_col] / df_combined[real_col]

# --- 4. Save to Excel ---
with pd.ExcelWriter(excel_file, engine="openpyxl") as writer:
    df_combined.to_excel(writer, sheet_name="GDP Data")
```

```
    df_deflator.to_excel(writer, sheet_name="GDP Deflator")

print(f"File '{excel_file}' created successfully.")

# --- 5. Add chart in Excel ---
wb = load_workbook(excel_file)
ws = wb["GDP Deflator"]

chart = LineChart()
chart.title = "GDP Deflators (Nominal / Real, Reference Year 2020)"
chart.style = 10
chart.y_axis.title = "Ratio (Nominal / Real)"
chart.x_axis.title = "Year"

# Define the data range for all deflators
min_col = 2
max_col = len(countries) + 1
min_row = 1
max_row = ws.max_row

data = Reference(ws, min_col=min_col, min_row=min_row, max_col=max_col,
max_row=max_row)
chart.add_data(data, titles_from_data=True)

# Define x-axis (years)
cats = Reference(ws, min_col=1, min_row=2, max_row=max_row)
```

```
chart.set_categories(cats)


# Place chart in the sheet
ws.add_chart(chart, "H3")


wb.save(excel_file)
print("Chart embedded successfully in 'GDP Deflator' sheet.")
print("Done.")
```