Data Preprocessing (Part 1)

Objective

The goal of this notebook is to clean and prepare the raw data for our analysis. We are working with three main data sources:

1. **Weather Data:** Rainfall, Solar Radiation, and Temperature.
2. **Crop Yield Data:** Agricultural output (our target variable).
3. **Farming Inputs:** Pesticide and Fertilizer usage.

We will convert the data from "wide" format (where months are columns) to "long" format (where each row represents a specific date) and standardize the country names and dates.

1. Setup and Library Imports

We import the necessary libraries to handle dataframes, mathematical operations, and file paths.

In [34]:
```
import pandas as pd
import numpy as np
import os
from datetime import date
```

2. Locate Raw Data Files

We list all the CSV files located in the precipitation, solar radiation, and temperature folders. We will loop through these lists later to process every file.

In [35]:
```
# List all CSV files in the data folders
precipitation_files = sorted(os.listdir("Data/precipitation_csv/"))
solar_files = sorted(os.listdir("Data/solar_radiation_csv/"))
temp_files = sorted(os.listdir("Data/temperature_csv/"))
```

3. Define Data Cleaning Functions

We define three functions to handle the specific formatting of the weather files. Since the raw data formats are similar, the logic for all three functions is consistent:

1. **Read the file:** Load the CSV and skip the header rows.
2. **Extract Country:** Get the country name directly from the filename.
3. **Reshape Data:** Convert the monthly columns (Jan-Dec) into rows using the `melt` function.
4. **Clean Dates:** Convert month names to numbers and create a standard Date object.
5. **Handle Missing Values:** Replace `-999` (error code) with `NaN` (Not a Number).

In [36]:
```python
def prep_rain(f):
    # Load file and parse country name from filename
    df = pd.read_csv(f"Data/precipitation_csv/{f}", header=10)
    country = "_".join(f.split("_")[:-6])

    # Select only relevant columns
    df['AREA'] = country
    df = df[['AREA','YEAR',
    'JAN', 'FEB', 'MAR',
    'APR', 'MAY', 'JUN',
    'JUL', 'AUG', 'SEP',
    'OCT', 'NOV', 'DEC',
    'ANN']]

    month_cols = ['JAN', 'FEB', 'MAR',
    'APR', 'MAY', 'JUN',
    'JUL', 'AUG', 'SEP',
    'OCT', 'NOV', 'DEC']

    # Melt into long format (Months become rows)
    df_long = df.melt(
        id_vars=["YEAR", "AREA"],
        value_vars=month_cols,
        var_name="MONTH",
        value_name="SUM"
    )

    # Clean month name: "jan_sum" -> "jan"
    df_long["MONTH"] = df_long["MONTH"].map(lambda x: x.split("_")[0])

    # Convert month abbreviations to numbers
    month_map = {
        "JAN": 1, "FEB": 2, "MAR": 3, "APR": 4, "MAY": 5, "JUN": 6,
        "JUL": 7, "AUG": 8, "SEP": 9, "OCT": 10, "NOV": 11, "DEC": 12
    }
    df_long["MONTH"] = df_long["MONTH"].map(month_map)

    # Sort data
    df_long = df_long.sort_values(["AREA", "YEAR", "MONTH"])
    df_long = df_long.dropna()

    # Create a proper datetime column
    df_long["MONTH"] = df_long["MONTH"].astype(int).astype(str).str.zfill(2)
    df_long["YEAR"] = df_long["YEAR"].astype(str)
    df_long["DATE"] = pd.to_datetime(df_long["YEAR"] + "-" + df_long["MONTH"]) +

    # Rename and clean values
    df_long.columns = ['year', 'area', 'month', 'rain', 'date']
    df_long["rain"] = df_long["rain"].replace(-999, np.nan)

    return df_long.reset_index(drop=True)
```

In [37]:
```python
def prep_solar(f):
    # Load file and parse country name from filename
    df = pd.read_csv(f"Data/solar_radiation_csv/{f}", header=10)
    country = "_".join(f.split("_")[:-8])

    df['AREA'] = country
    df = df[['AREA','YEAR',
    'JAN', 'FEB', 'MAR',
    'APR', 'MAY', 'JUN',
    'JUL', 'AUG', 'SEP',
    'OCT', 'NOV', 'DEC',
    'ANN']]

    month_cols = ['JAN', 'FEB', 'MAR',
    'APR', 'MAY', 'JUN',
    'JUL', 'AUG', 'SEP',
    'OCT', 'NOV', 'DEC']

    # Melt into long format
    df_long = df.melt(
        id_vars=["YEAR", "AREA"],
        value_vars=month_cols,
        var_name="MONTH",
        value_name="SUM"
    )

    # Clean month name
    df_long["MONTH"] = df_long["MONTH"].map(lambda x: x.split("_")[0])

    # Convert month abbreviations to numbers
    month_map = {
        "JAN": 1, "FEB": 2, "MAR": 3, "APR": 4, "MAY": 5, "JUN": 6,
        "JUL": 7, "AUG": 8, "SEP": 9, "OCT": 10, "NOV": 11, "DEC": 12
    }
    df_long["MONTH"] = df_long["MONTH"].map(month_map)

    # Sort data
    df_long = df_long.sort_values(["AREA", "YEAR", "MONTH"])
    df_long = df_long.dropna()

    # Create a proper datetime column
    df_long["MONTH"] = df_long["MONTH"].astype(int).astype(str).str.zfill(2)
    df_long["YEAR"] = df_long["YEAR"].astype(str)
    df_long["DATE"] = pd.to_datetime(df_long["YEAR"] + "-" + df_long["MONTH"]) +

    # Rename and clean values
    df_long.columns = ['year', 'area', 'month', 'solar', 'date']
    df_long["solar"] = df_long["solar"].replace(-999, np.nan)

    return df_long.reset_index(drop=True)
```

In [38]:
```python
def prep_temp(f):
    # Load file and parse country name from filename
    df = pd.read_csv(f"Data/temperature_csv/{f}", header=10)
    country = "_".join(f.split("_")[:-5])

    df['AREA'] = country
    df = df[['AREA','YEAR',
    'JAN', 'FEB', 'MAR',
    'APR', 'MAY', 'JUN',
    'JUL', 'AUG', 'SEP',
    'OCT', 'NOV', 'DEC',
    'ANN']]

    month_cols = ['JAN', 'FEB', 'MAR',
    'APR', 'MAY', 'JUN',
    'JUL', 'AUG', 'SEP',
    'OCT', 'NOV', 'DEC',
    'ANN']

    # Melt into long format
    df_long = df.melt(
        id_vars=["YEAR", "AREA"],
        value_vars=month_cols,
        var_name="MONTH",
        value_name="SUM"
    )

    # Clean month name
    df_long["MONTH"] = df_long["MONTH"].map(lambda x: x.split("_")[0])

    # Convert month abbreviations to numbers
    month_map = {
        "JAN": 1, "FEB": 2, "MAR": 3, "APR": 4, "MAY": 5, "JUN": 6,
        "JUL": 7, "AUG": 8, "SEP": 9, "OCT": 10, "NOV": 11, "DEC": 12
    }
    df_long["MONTH"] = df_long["MONTH"].map(month_map)

    # Sort data
    df_long = df_long.sort_values(["AREA", "YEAR", "MONTH"])
    df_long = df_long.dropna()

    # Create a proper datetime column
    df_long["MONTH"] = df_long["MONTH"].astype(int).astype(str).str.zfill(2)
    df_long["YEAR"] = df_long["YEAR"].astype(str)
    df_long["DATE"] = pd.to_datetime(df_long["YEAR"] + "-" + df_long["MONTH"]) +

    # Rename and clean values
    df_long.columns = ['year', 'area', 'month', 'temp', 'date']
    df_long["temp"] = df_long["temp"].replace(-999, np.nan)

    return df_long.reset_index(drop=True)
```

4. Process and Merge Weather Data

Now that our functions are defined, we apply them to all the files in our lists. We then combine (concatenate) the results into three main dataframes: one for rain, one for solar radiation, and one for temperature.

In [39]:

```
# Run the processing functions on all listed files
rain_data = pd.concat([prep_rain(f) for f in precipitation_files])
solar_data = pd.concat([prep_solar(f) for f in solar_files])
temp_data = pd.concat([prep_temp(f) for f in temp_files])
```

5. Create Final Weather Dataset

We merge the three weather datasets into a single dataframe called `nasa_df`. We join them based on `year`, `area`, `date`, and `month` to ensure the rows align correctly. Finally, we save this clean dataset as a Parquet file.

In [40]:

```
# Merge Rain, Solar, and Temp based on Date and Area
nasa_df = rain_data.merge(
    solar_data, on=['year', 'area', 'date', 'month']
).merge(
    temp_data, on=['year', 'area', 'date', 'month']
)
```

In [41]:
```
# Select final columns and save to Parquet format
nasa_df = nasa_df[['date', 'area', 'rain', 'solar', 'temp']]
nasa_df.to_parquet('Parquet/nasa_df.parquet')
```

In [42]:
```
# Display the result
nasa_df
```

Out[42]:

	date	area	rain	solar	temp
0	1981-01-31	Afghanistan	55.53	NaN	-0.95
1	1981-02-28	Afghanistan	85.20	NaN	0.97
2	1981-03-31	Afghanistan	66.13	NaN	6.18
3	1981-04-30	Afghanistan	23.64	NaN	13.07
4	1981-05-31	Afghanistan	23.92	NaN	17.61
...	...	...	...	...	...
107839	2023-08-31	Zimbabwe	0.10	19.47	18.23
107840	2023-09-30	Zimbabwe	2.65	22.55	23.64
107841	2023-10-31	Zimbabwe	137.78	23.96	24.80
107842	2023-11-30	Zimbabwe	27.16	26.16	26.37
107843	2023-12-31	Zimbabwe	171.60	22.59	24.48

107844 rows × 5 columns

6. Process Crop Yield Data

In this section, we prepare the target variable (crop yield).

1. We load the raw CSV file.
2. We clean the column names for consistency.
3. We standardize the `Area` names (replacing spaces with underscores).
4. We format the `Year` column to be a full date object (set to December 31st of that year).

In [43]:
```
# Load raw yield data
yield_crop = pd.read_csv('Data/yield_final.csv')

# Select relevant columns
yield_crop = yield_crop[['Area', 'Item', 'Year', 'Yield (kg/ha)']]

# Rename columns to lower case
yield_crop.columns = ['area', 'item', 'year', 'label']
```

In [44]:
```
# Clean Area names (replace spaces with underscores)
yield_crop['area'] = yield_crop['area'].str.replace(' ', '_')
```

```
# Convert Year to a full date (set to end of year)
yield_crop['year'] = yield_crop['year'].map(lambda x: date(int(x), 12, 31))
```

In [45]:
```
# Save processed yield data
yield_crop.to_parquet('Parquet/label_yield.parquet')
```

In [46]:
```
# Display the result
yield_crop
```

Out[46]:

	area	item	year	label
0	Afghanistan	Maize (corn)	1970-12-31	1475.7
1	Afghanistan	Maize (corn)	1971-12-31	1340.0
2	Afghanistan	Maize (corn)	1972-12-31	1565.2
3	Afghanistan	Maize (corn)	1973-12-31	1617.0
4	Afghanistan	Maize (corn)	1974-12-31	1617.0
...	...	...	...	...
89255	Zimbabwe	Watermelons	2019-12-31	25000.0
89256	Zimbabwe	Watermelons	2020-12-31	36000.0
89257	Zimbabwe	Watermelons	2021-12-31	31377.0
89258	Zimbabwe	Watermelons	2022-12-31	33841.3
89259	Zimbabwe	Watermelons	2023-12-31	32668.1

89260 rows × 4 columns

7. Process Pesticides & Fertilizers

We incorporate data on farming inputs, which are key features for our model.

Steps taken:

1. **Load Data:** Read the pesticide and fertilizer CSV files.
2. **Merge:** Combine them into one dataframe based on Country and Year.
3. **Standardize Country Names:** The raw data uses different spellings for countries (e.g., "Turkey" vs. "Turkiye"). We apply a dictionary mapping to ensure these names match our other datasets.
4. **Format Dates:** Convert the year into a standardized date format.
5. **Save:** Export the final cleaned dataframe to Parquet.

In [47]:
```
import pandas as pd
from datetime import date

# 1. Load the datasets
pesticides_df = pd.read_csv('Data/pesticides.csv')
fertilizer_df = pd.read_csv('Data/fertilizer.csv')
```

```python
# 2. Merge them on 'Area' and 'Year'
# Using outer join to keep all records from both files
farming_df = pd.merge(pesticides_df, fertilizer_df, on=['Area', 'Year'], how='ou

# 3. Define the renaming dictionary based on your standard list
area_mapping = {
    "Bolivia": "Bolivia (Plurinational State of)",
    "Congo, Dem. Rep.": "Democratic Republic of the Congo",
    "Congo, Rep.": "Congo",
    "Cote d'Ivoire": "Côte d'Ivoire",
    "Gambia, The": "Gambia",
    "Hong Kong SAR, China": "China, Hong Kong SAR",
    "Iran, Islamic Rep.": "Iran (Islamic Republic of)",
    "Korea, Dem. People's Rep.": "Democratic People's Republic of Korea",
    "Korea, Rep.": "Republic of Korea",
    "Kyrgyz Republic": "Kyrgyzstan",
    "Lao PDR": "Lao People's Democratic Republic",
    "Micronesia, Fed. Sts.": "Micronesia (Federated States of)",
    "Moldova": "Republic of Moldova",
    "Netherlands": "Netherlands (Kingdom of the)",
    "St. Kitts and Nevis": "Saint Kitts and Nevis",
    "St. Lucia": "Saint Lucia",
    "St. Vincent and the Grenadines": "Saint Vincent and the Grenadines",
    "Slovak Republic": "Slovakia",
    "Taiwan, China": "China, Taiwan Province of",
    "Tanzania": "United Republic of Tanzania",
    "Turkey": "Türkiye",
    "Turkiye": "Türkiye",
    "United Kingdom": "United Kingdom of Great Britain and Northern Ireland",
    "United States": "United States of America",
    "Venezuela, RB": "Venezuela (Bolivarian Republic of)",
    "West Bank and Gaza": "Palestine",
    "Yemen, Rep.": "Yemen"
}

# 4. Apply the renaming
farming_df['Area'] = farming_df['Area'].replace(area_mapping)

# 5. Convert Year to a full date (set to end of year)
# Drop rows with missing years (if any) and ensure integer type
farming_df = farming_df.dropna(subset=['Year'])
farming_df['Year'] = farming_df['Year'].astype(int)
farming_df['Year'] = farming_df['Year'].map(lambda x: date(int(x), 12, 31))

# 6. Standardize column names to lowercase
farming_df.columns = farming_df.columns.str.lower()

# 7. Save to Parquet
farming_df.to_parquet('Parquet/farming_df.parquet', index=False)

# Verification
print("Shape:", farming_df.shape)
print(farming_df.head())
```

```
Shape: (14977, 4)
          area        year  pesticides  fertilizer
0  Afghanistan  1970-12-31         NaN    2.465057
1  Afghanistan  1971-12-31         NaN    2.594937
2  Afghanistan  1972-12-31         NaN    3.680152
3  Afghanistan  1973-12-31         NaN    3.109987
4  Afghanistan  1974-12-31         NaN    4.285714
```