20250429_01

April 29, 2025

[245]:
```
import pandas as pd

data = pd.read_csv('data.csv')
```

[247]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29 entries, 0 to 28
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Country Name    26 non-null     object
 1   Country Code    24 non-null     object
 2   Series Name     24 non-null     object
 3   Series Code     24 non-null     object
 4   2016 [YR2016]   24 non-null     float64
 5   2017 [YR2017]   24 non-null     float64
 6   2018 [YR2018]   24 non-null     float64
 7   2019 [YR2019]   24 non-null     float64
 8   2020 [YR2020]   24 non-null     object
dtypes: float64(4), object(5)
memory usage: 2.2+ KB
```

0.0.1 Structure Overview

- **Columns:** 9 total — `Country Name`, `Country Code`, `Series Name`, `Series Code`, and data from 2016 to 2020.
- **Rows:** 29 total, but only 24 expected (8 countries × 3 indicators).
- **Missing Values:**
  - `Country Name`: 26 non-null → 2 additional unexpected values
  - All other key columns have 24 non-null values → matches expectation ### Observations
- **Country Name, Country Code, etc**: Correctly typed as `object`.
- **2016 – 2019**: Correctly typed as `float64`.
- **2020**: Typed as `object`, likely due to presence of empty strings or non-numeric text.

[250]:
```
data.head(29)
```

[250]:

```
                                          Country Name Country Code  \
0                                            Argentina          ARG
1                                            Argentina          ARG
2                                            Argentina          ARG
3                                               Brazil          BRA
4                                               Brazil          BRA
5                                               Brazil          BRA
6                                                Chile          CHL
7                                                Chile          CHL
8                                                Chile          CHL
9                                             Paraguay          PRY
10                                            Paraguay          PRY
11                                            Paraguay          PRY
12                                             Uruguay          URY
13                                             Uruguay          URY
14                                             Uruguay          URY
15                                             Bolivia          BOL
16                                             Bolivia          BOL
17                                             Bolivia          BOL
18                                                Peru          PER
19                                                Peru          PER
20                                                Peru          PER
21                                             Ecuador          ECU
22                                             Ecuador          ECU
23                                             Ecuador          ECU
24                                                 NaN          NaN
25                                                 NaN          NaN
26                                                 NaN          NaN
27  Data from database: Sustainable Development Go…          NaN
28                            Last Updated: 07/22/2022          NaN

                                 Series Name     Series Code  2016 [YR2016]  \
0   Access to electricity (% of population)  EG.ELC.ACCS.ZS   9.984958e+01
1    CO2 emissions (metric tons per capita)  EN.ATM.CO2E.PC   4.201846e+00
2                  GDP (constant 2015 US$)  NY.GDP.MKTP.KD   5.823766e+11
3   Access to electricity (% of population)  EG.ELC.ACCS.ZS   9.970000e+01
4    CO2 emissions (metric tons per capita)  EN.ATM.CO2E.PC   2.168575e+00
5                  GDP (constant 2015 US$)  NY.GDP.MKTP.KD   1.743173e+12
6   Access to electricity (% of population)  EG.ELC.ACCS.ZS   1.000000e+02
7    CO2 emissions (metric tons per capita)  EN.ATM.CO2E.PC   4.749830e+00
8                  GDP (constant 2015 US$)  NY.GDP.MKTP.KD   2.467477e+11
9   Access to electricity (% of population)  EG.ELC.ACCS.ZS   9.840000e+01
10   CO2 emissions (metric tons per capita)  EN.ATM.CO2E.PC   1.059329e+00
11                 GDP (constant 2015 US$)  NY.GDP.MKTP.KD   3.775688e+10
12  Access to electricity (% of population)  EG.ELC.ACCS.ZS   9.970000e+01
13   CO2 emissions (metric tons per capita)  EN.ATM.CO2E.PC   1.904128e+00
14                 GDP (constant 2015 US$)  NY.GDP.MKTP.KD   5.417453e+10
```

15	Access to electricity (% of population)	EG.ELC.ACCS.ZS	9.180000e+01
16	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	1.995137e+00
17	GDP (constant 2015 US$)	NY.GDP.MKTP.KD	3.440730e+10
18	Access to electricity (% of population)	EG.ELC.ACCS.ZS	9.420000e+01
19	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	1.838580e+00
20	GDP (constant 2015 US$)	NY.GDP.MKTP.KD	1.973089e+11
21	Access to electricity (% of population)	EG.ELC.ACCS.ZS	9.870000e+01
22	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	2.414027e+00
23	GDP (constant 2015 US$)	NY.GDP.MKTP.KD	9.807270e+10
24	NaN	NaN	NaN
25	NaN	NaN	NaN
26	NaN	NaN	NaN
27	NaN	NaN	NaN
28	NaN	NaN	NaN

	2017 [YR2017]	2018 [YR2018]	2019 [YR2019]	2020 [YR2020]
0	1.000000e+02	9.998958e+01	1.000000e+02	100
1	4.071308e+00	3.975772e+00	3.740650e+00	..
2	5.987909e+11	5.831181e+11	5.713045e+11	514772410744.886
3	9.980000e+01	9.970000e+01	9.980000e+01	100
4	2.196418e+00	2.071855e+00	2.057811e+00	..
5	1.766233e+12	1.797737e+12	1.819683e+12	1749103394213.21
6	9.970000e+01	1.000000e+02	1.000000e+02	100
7	4.714020e+00	4.624338e+00	4.821118e+00	..
8	2.500978e+11	2.600768e+11	2.620808e+11	246412987238.941
9	9.930000e+01	9.960000e+01	9.970000e+01	100
10	1.173720e+00	1.217642e+00	1.165425e+00	..
11	3.957302e+10	4.084104e+10	4.067692e+10	40343452707.5908
12	9.980000e+01	9.980000e+01	9.990000e+01	100
13	1.774987e+00	1.896042e+00	1.874785e+00	..
14	5.505636e+10	5.531948e+10	5.551334e+10	52115108174.7165
15	9.180000e+01	9.280000e+01	9.508000e+01	97.5541229248047
16	2.032547e+00	2.046130e+00	1.940398e+00	..
17	3.585076e+10	3.736496e+10	3.819323e+10	34855949803.3644
18	9.480000e+01	9.520000e+01	9.555136e+01	99.3118133544922
19	1.725909e+00	1.706510e+00	1.745592e+00	..
20	2.022788e+11	2.103080e+11	2.150202e+11	191469666109.311
21	9.920000e+01	9.870000e+01	9.909000e+01	98.8499984741211
22	2.296645e+00	2.349517e+00	2.261470e+00	..
23	1.003954e+11	1.016898e+11	1.017021e+11	93781977159.7812
24	NaN	NaN	NaN	NaN
25	NaN	NaN	NaN	NaN
26	NaN	NaN	NaN	NaN
27	NaN	NaN	NaN	NaN
28	NaN	NaN	NaN	NaN

0.0.2 Observations

1. `Country Name` and `Country Code` are redundant — keep one.
2. `Series Name` and `Series Code` are also redundant — keep one.
3. As shown in **2020[YR2020]**, e.g., All CO emissions are blank, but there are also float and NaN → **2020** is typed as `object`
4. Indicators have different scales:
   - Electricity access: max ~100
   - CO emissions: small values (single digit)
   - GDP: large absolute values → **may require standardization**

[253]:
```
data.describe()
```

[253]:

	2016 [YR2016]	2017 [YR2017]	2018 [YR2018]	2019 [YR2019]
count	2.400000e+01	2.400000e+01	2.400000e+01	2.400000e+01
mean	1.247507e+11	1.270115e+11	1.286023e+11	1.293406e+11
std	3.682678e+11	3.736070e+11	3.789742e+11	3.826451e+11
min	1.059329e+00	1.173720e+00	1.217642e+00	1.165425e+00
25%	3.754891e+00	3.627642e+00	3.569208e+00	3.370855e+00
50%	9.920000e+01	9.950000e+01	9.965000e+01	9.975000e+01
75%	4.186130e+10	4.344385e+10	4.446065e+10	4.438602e+10
max	1.743173e+12	1.766233e+12	1.797737e+12	1.819683e+12

0.0.3 Observations

1. `.describe()` not useful due to inconsistent units and missing values.

Next step: filter valid rows, fix 2020 column type, and restructure data.

[256]:
```
# First we filter valid rows with Country Name
data['Country Name'].unique()
```

[256]:
```
array(['Argentina', 'Brazil', 'Chile', 'Paraguay', 'Uruguay', 'Bolivia',
       'Peru', 'Ecuador', nan,
       'Data from database: Sustainable Development Goals (SDGs)',
       'Last Updated: 07/22/2022'], dtype=object)
```

[258]:
```
target_countries = ['Argentina', 'Brazil', 'Chile', 'Paraguay', 'Uruguay',␣
  ↪'Bolivia', 'Peru', 'Ecuador']
data = data[data['Country Name'].isin(target_countries)]
```

[260]:
```
# Checking if the cleaning is OK
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 24 entries, 0 to 23
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
```

```
---  ------          --------------  -----
 0   Country Name    24 non-null     object
 1   Country Code    24 non-null     object
 2   Series Name     24 non-null     object
 3   Series Code     24 non-null     object
 4   2016 [YR2016]   24 non-null     float64
 5   2017 [YR2017]   24 non-null     float64
 6   2018 [YR2018]   24 non-null     float64
 7   2019 [YR2019]   24 non-null     float64
 8   2020 [YR2020]   24 non-null     object
dtypes: float64(4), object(5)
memory usage: 1.9+ KB
```

Looking good, we shall proceed.

[263]:
```
data['Series Name'].unique()
```

[263]:
```
array(['Access to electricity (% of population)',
       'CO2 emissions (metric tons per capita)',
       'GDP (constant 2015 US$)'], dtype=object)
```

No need to clean, nice. Now we drop some redundant columns.

[266]:
```
data = data.drop(columns = ['Country Code', 'Series Code'])
```

[268]:
```
# Checking again
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 24 entries, 0 to 23
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Country Name    24 non-null     object
 1   Series Name     24 non-null     object
 2   2016 [YR2016]   24 non-null     float64
 3   2017 [YR2017]   24 non-null     float64
 4   2018 [YR2018]   24 non-null     float64
 5   2019 [YR2019]   24 non-null     float64
 6   2020 [YR2020]   24 non-null     object
dtypes: float64(4), object(3)
memory usage: 1.5+ KB
```

Looking good, yipee. Now we clean 2020

[271]:
```
data['2020 [YR2020]'] = pd.to_numeric(data['2020 [YR2020]'], errors = 'coerce')
```

[273]:
```
# Checking again
data['2020 [YR2020]'].dtype
```

[273]: dtype('float64')

[275]:
```
data['2020 [YR2020]'].isna().sum()
```

[275]: 8

Now all blanks in 2020 are NaN. I choose to just keep them for now.

Next I make the data easier to read.

[298]:
```
# Step 1: Making it tidy
data_melted = data.melt(id_vars = ['Country Name', 'Series Name'],
                        value_vars = ['2016 [YR2016]', '2017 [YR2017]', '2018␣
↪[YR2018]', '2019 [YR2019]', '2020 [YR2020]'],
                        var_name = 'year', value_name = 'value')

# Step 2: Rearranging
data_final = data_melted.pivot_table(index = ['Country Name', 'year'],
                                     columns = 'Series Name',
                                     values = 'value').reset_index()
```

[300]:
```
# Check
data_final.head()
```

[300]:
```
Series Name Country Name           year  \
0              Argentina  2016 [YR2016]
1              Argentina  2017 [YR2017]
2              Argentina  2018 [YR2018]
3              Argentina  2019 [YR2019]
4              Argentina  2020 [YR2020]

Series Name  Access to electricity (% of population)  \
0                                          99.849579
1                                         100.000000
2                                          99.989578
3                                         100.000000
4                                         100.000000

Series Name  CO2 emissions (metric tons per capita)  GDP (constant 2015 US$)
0                                          4.201846             5.823766e+11
1                                          4.071308             5.987909e+11
2                                          3.975772             5.831181e+11
3                                          3.740650             5.713045e+11
4                                               NaN             5.147724e+11
```

[302]:
```
# Exporting the data
data_final.to_csv('cleaned_data.csv', index = False)
```