Day17_Pandas_Introduction_with_GDP_Dataset1

June 5, 2025

Day 17 – Pandas Lab Introduction with GDP Mini Project

Today I explored **Pandas**, one of the most powerful Python libraries for working with structured/tabular data. It makes data cleaning, filtering, analysis, and visualization much easier and more efficient.

What is Pandas?

Pandas is a powerful open-source Python library used for **data manipulation, analysis, and cleaning**.
The name "Pandas" is derived from **"Panel Data"** – an econometrics term for multidimensional structured data.

It provides **data structures** like: - **Series**: One-dimensional labeled array. - **DataFrame**: Two-dimensional labeled data structure (like a table with rows and columns).

> **Pandas = Works with DataFrames**
> **NumPy = Works with Arrays**

Real-Life Use Case

Imagine a **company with a database (DB)**: - It has **multiple tables**: `customers`, `sales`, `inventory`, `employees`. - Each table has **lots of rows and columns**. - Pandas helps **load, filter, analyze, and visualize** this tabular data easily — just like handling Excel but more powerful and programmable.

Pandas is the best tool when: - You are dealing with **CSV files**, **Excel sheets**, or **SQL databases**. - You want to **analyze trends**, **clean messy data**, or **merge datasets**.

Where is Pandas Used?

- Data Science and Machine Learning
- Finance and Accounting
- Healthcare (patient records)
- Business Reporting and Analytics
- E-commerce (user data, product data)
- Government data analysis (like GDP per country)

Advantages of Pandas

- Easy to use and write code
- Fast filtering and data aggregation
- Built-in functions for reading/writing CSV, Excel, SQL
- Supports missing data handling (`NaN`)
- Seamless integration with NumPy, Matplotlib, Seaborn, Scikit-learn
- Enables pivot tables, group-by summaries, merging datasets

Disadvantages of Pandas

- Slower with **very large datasets** (millions of rows) – better to use **Dask** or **PySpark** then
- Uses more memory compared to optimized libraries like NumPy
- Can be complex for beginners when chaining too many operations

---

Key Connections

Library	Purpose
Pandas	DataFrames and tabular data
NumPy	Arrays and numerical operations
Matplotlib	General data visualization
Seaborn	Statistical data visualization

Understanding Excel Sheets as Datasets

In the world of data analysis and machine learning, we often refer to an **Excel sheet** or a **CSV file** as a **dataset**.

Dataset Structure

Concept	Meaning in Excel / Pandas
Dataset	An entire Excel sheet or CSV file
Columns	Features / Attributes / Variables
Rows	Records / Observations / Data points

Explanation: - **Columns** represent the **type of data** we are collecting (e.g., Country, GDP, Population).
These are also called: - **Attributes** in databases - **Features** in machine learning - **Variables** in statistics

- **Rows** represent **individual entries** in the dataset.
  Each row is a **record**, an **observation**, or a **data point**.

For example: If we are analyzing GDP data:

Country	Year	GDP (in Trillion USD)
India	2023	3.73

Country	Year	GDP (in Trillion USD)
Germany	2023	4.42
USA	2023	26.95

Here: - Each row = one **record** - Each column = one **feature/attribute**

[1]:
```
# Import pandas lab to use
import pandas as pd
```

[2]:
```
# Check version
pd.__version__
```

[2]:
```
'2.2.2'
```

1 csv_overview functions

[4]:
```
# Reading the GDP dataset using Pandas
# 'r' before the file path indicates a raw string, so that backslashes (\) are␣
 ↪treated literally
# pd.read_csv() is a Pandas function used to load a CSV (Comma-Separated␣
 ↪Values) file as a DataFrame
# In this case, we are reading the GDP dataset located on the Desktop into a␣
 ↪DataFrame called 'df'
# Load data
df = pd.read_csv(r'C:
 ↪\Users\aksha\OneDrive\Desktop\Dataset\Pandas_intro_gdp_data\data.csv')
```

[5]:
```
df
```

[5]:
```
              CountryName CountryCode  BirthRate  InternetUsers  \
0                   Aruba         ABW     10.244           78.9
1             Afghanistan         AFG     35.253            5.9
2                  Angola         AGO     45.985           19.1
3                 Albania         ALB     12.877           57.2
4    United Arab Emirates         ARE     11.044           88.0
..                    …           …          …              …
190          Yemen, Rep.         YEM     32.947           20.0
191         South Africa         ZAF     20.850           46.5
192     Congo, Dem. Rep.         COD     42.394            2.2
193               Zambia         ZMB     40.471           15.4
194             Zimbabwe         ZWE     35.715           18.5

             IncomeGroup
0            High income
1             Low income
2    Upper middle income
```

```
3    Upper middle income
4             High income
..                    …
190  Lower middle income
191  Upper middle income
192           Low income
193  Lower middle income
194           Low income

[195 rows x 5 columns]
```

[6]:
```
id(df)
```

[6]:
```
2543758601936
```

[7]:
```
type(df)
```

[7]:
```
pandas.core.frame.DataFrame
```

1.1 Shape of Dataset / How many rows & columns

[9]:
```
df.shape # Give output as total (row_count,column_count )
```

[9]:
```
(195, 5)
```

1.2 Row Count

[8]:
```
len(df) #row count
```

[8]:
```
195
```

1.3 Columns Count

[10]:
```
df.columns # Print all columns names
```

[10]:
```
Index(['CountryName', 'CountryCode', 'BirthRate', 'InternetUsers',
       'IncomeGroup'],
      dtype='object')
```

[11]:
```
len(df.columns) # find  column count
```

[11]:
```
5
```

1.4 Missing values & total counts

[12]:
```
df.isnull() # To check missing values , also write as df.isna()
```

[12]:
```
     CountryName  CountryCode  BirthRate  InternetUsers  IncomeGroup
0          False        False      False          False        False
1          False        False      False          False        False
2          False        False      False          False        False
3          False        False      False          False        False
4          False        False      False          False        False
..           …            …          …              …            …
190        False        False      False          False        False
191        False        False      False          False        False
192        False        False      False          False        False
193        False        False      False          False        False
194        False        False      False          False        False

[195 rows x 5 columns]
```

[13]:
```
df.isna() # other way to write data_frame.isnull()
```

[13]:
```
     CountryName  CountryCode  BirthRate  InternetUsers  IncomeGroup
0          False        False      False          False        False
1          False        False      False          False        False
2          False        False      False          False        False
3          False        False      False          False        False
4          False        False      False          False        False
..           …            …          …              …            …
190        False        False      False          False        False
191        False        False      False          False        False
192        False        False      False          False        False
193        False        False      False          False        False
194        False        False      False          False        False

[195 rows x 5 columns]
```

[14]:
```
df.isnull().sum() # Gives output as total numbers of missing / null values else␣
↪0 for not
```

[14]:
```
CountryName      0
CountryCode      0
BirthRate        0
InternetUsers    0
IncomeGroup      0
dtype: int64
```

1.5 Head & Tail

[15]:
```
df.head() # top 5 rows by default
```

[15]:
```
            CountryName CountryCode  BirthRate  InternetUsers  \
0                 Aruba         ABW     10.244           78.9
1           Afghanistan         AFG     35.253            5.9
2                Angola         AGO     45.985           19.1
3               Albania         ALB     12.877           57.2
4  United Arab Emirates         ARE     11.044           88.0

           IncomeGroup
0          High income
1           Low income
2  Upper middle income
3  Upper middle income
4          High income
```

[16]:
```
df.tail() # last 5 rows by default
```

[16]:
```
          CountryName CountryCode  BirthRate  InternetUsers  \
190       Yemen, Rep.         YEM     32.947           20.0
191      South Africa         ZAF     20.850           46.5
192  Congo, Dem. Rep.         COD     42.394            2.2
193            Zambia         ZMB     40.471           15.4
194          Zimbabwe         ZWE     35.715           18.5

             IncomeGroup
190  Lower middle income
191  Upper middle income
192           Low income
193  Lower middle income
194           Low income
```

[17]:
```
# Return n number of rows (head = start)
df.head(3)
```

[17]:
```
   CountryName CountryCode  BirthRate  InternetUsers          IncomeGroup
0        Aruba         ABW     10.244           78.9          High income
1  Afghanistan         AFG     35.253            5.9           Low income
2       Angola         AGO     45.985           19.1  Upper middle income
```

[25]:
```
# Return n number of rows (tail = bottom)
df.tail(13)
```

[25]:
```
                       CountryName CountryCode  BirthRate  InternetUsers  \
182                     Uzbekistan         UZB     22.500           38.2
183  St. Vincent and the Grenadines         VCT     16.306           52.0
184                  Venezuela, RB         VEN     19.842           54.9
185          Virgin Islands (U.S.)         VIR     10.700           45.3
186                        Vietnam         VNM     15.537           43.9
187                        Vanuatu         VUT     26.739           11.3
```

```
188              West Bank and Gaza         PSE     30.394           46.6
189                           Samoa         WSM     26.172           15.3
190                     Yemen, Rep.         YEM     32.947           20.0
191                    South Africa         ZAF     20.850           46.5
192                Congo, Dem. Rep.         COD     42.394            2.2
193                          Zambia         ZMB     40.471           15.4
194                        Zimbabwe         ZWE     35.715           18.5

             IncomeGroup
182  Lower middle income
183  Upper middle income
184          High income
185          High income
186  Lower middle income
187  Lower middle income
188  Lower middle income
189  Lower middle income
190  Lower middle income
191  Upper middle income
192           Low income
193  Lower middle income
194           Low income
```

1.6 Select Random/Sample Rows

[23]:
```
#Retrun n number of smaple rows accross data (any random rows)
df.sample(n=5, replace=True)
```

[23]:
```
       CountryName CountryCode  BirthRate  InternetUsers           IncomeGroup
172    Timor-Leste         TLS     35.755            1.1  Lower middle income
168           Togo         TGO     36.080            4.5           Low income
177       Tanzania         TZA     39.518            4.4           Low income
184  Venezuela, RB         VEN     19.842           54.9          High income
182     Uzbekistan         UZB     22.500           38.2  Lower middle income
```

1.7 Data types & Information

[20]:
```
df.dtypes # Return only data type of each Column
```

[20]:
```
CountryName       object
CountryCode       object
BirthRate        float64
InternetUsers    float64
IncomeGroup       object
dtype: object
```

[22]:
```
# Return all information about range index, memory usage,
# count of null values, dataset  & data type
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   CountryName    195 non-null    object
 1   CountryCode    195 non-null    object
 2   BirthRate      195 non-null    float64
 3   InternetUsers  195 non-null    float64
 4   IncomeGroup    195 non-null    object
dtypes: float64(2), object(3)
memory usage: 7.7+ KB
```

1.8 Describe & Describe All

- **Describe (Numeric)**
- **Describe All (Numeric + Categorical)**

[41]:
```
df.describe() # Only Numerical data
```

[41]:

	BirthRate	InternetUsers
count	195.000000	195.000000
mean	21.469928	42.076471
std	10.605467	29.030788
min	7.900000	0.900000
25%	12.120500	14.520000
50%	19.680000	41.000000
75%	29.759500	66.225000
max	49.661000	96.546800

[42]:
```
df.describe(include='all') # Every Thing
```

[42]:

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup
count	195	195	195.000000	195.000000	195
unique	195	195	NaN	NaN	4
top	Aruba	ABW	NaN	NaN	High income
freq	1	1	NaN	NaN	67
mean	NaN	NaN	21.469928	42.076471	NaN
std	NaN	NaN	10.605467	29.030788	NaN
min	NaN	NaN	7.900000	0.900000	NaN
25%	NaN	NaN	12.120500	14.520000	NaN
50%	NaN	NaN	19.680000	41.000000	NaN
75%	NaN	NaN	29.759500	66.225000	NaN
max	NaN	NaN	49.661000	96.546800	NaN

2 Slicing in Data Frame

Quick Notes:

- df[ ] = rows when slicing, or column if string.
- df[::] = step slicing.
- .loc[ ] is label-based (use names).
- .iloc[ ] is index-based (use numbers).

[26]:
```
df[:]           # All rows (full DataFrame)
```

[26]:
```
              CountryName CountryCode  BirthRate  InternetUsers  \
0                   Aruba         ABW     10.244           78.9
1             Afghanistan         AFG     35.253            5.9
2                  Angola         AGO     45.985           19.1
3                 Albania         ALB     12.877           57.2
4    United Arab Emirates         ARE     11.044           88.0
..                    …           …          …              …
190          Yemen, Rep.         YEM     32.947           20.0
191         South Africa         ZAF     20.850           46.5
192     Congo, Dem. Rep.         COD     42.394            2.2
193               Zambia         ZMB     40.471           15.4
194             Zimbabwe         ZWE     35.715           18.5

             IncomeGroup
0            High income
1             Low income
2    Upper middle income
3    Upper middle income
4            High income
..                    …
190  Lower middle income
191  Upper middle income
192           Low income
193  Lower middle income
194           Low income

[195 rows x 5 columns]
```

[27]:
```
df[::-1]        # All rows in reverse order
```

[27]:
```
              CountryName CountryCode  BirthRate  InternetUsers  \
194             Zimbabwe         ZWE     35.715           18.5
193               Zambia         ZMB     40.471           15.4
192     Congo, Dem. Rep.         COD     42.394            2.2
191         South Africa         ZAF     20.850           46.5
190          Yemen, Rep.         YEM     32.947           20.0
```

```
..                    …          …         …              …
4    United Arab Emirates         ARE     11.044           88.0
3                 Albania         ALB     12.877           57.2
2                  Angola         AGO     45.985           19.1
1             Afghanistan         AFG     35.253            5.9
0                   Aruba         ABW     10.244           78.9

             IncomeGroup
194           Low income
193  Lower middle income
192           Low income
191  Upper middle income
190  Lower middle income
..                   …
4             High income
3    Upper middle income
2    Upper middle income
1             Low income
0             High income

[195 rows x 5 columns]
```

[28]:
```
df[:11]          # First 11 rows (row 0 to 10)
```

[28]:
```
             CountryName CountryCode  BirthRate  InternetUsers  \
0                  Aruba         ABW     10.244        78.9000
1            Afghanistan         AFG     35.253         5.9000
2                 Angola         AGO     45.985        19.1000
3                Albania         ALB     12.877        57.2000
4   United Arab Emirates         ARE     11.044        88.0000
5              Argentina         ARG     17.716        59.9000
6                Armenia         ARM     13.308        41.9000
7    Antigua and Barbuda         ATG     16.447        63.4000
8              Australia         AUS     13.200        83.0000
9                Austria         AUT      9.400        80.6188
10            Azerbaijan         AZE     18.300        58.7000

             IncomeGroup
0           High income
1            Low income
2   Upper middle income
3   Upper middle income
4           High income
5           High income
6   Lower middle income
7           High income
8           High income
```

```
9           High income
10  Upper middle income
```

[29]:
```
df[0:200:50]    # Rows from 0 to 199 with a step of 50 (rows 0, 50, 100, 150)
```

[29]:
```
    CountryName CountryCode  BirthRate  InternetUsers          IncomeGroup
0         Aruba         ABW     10.244      78.900000          High income
50      Ecuador         ECU     21.070      40.353684  Upper middle income
100       Libya         LBY     21.425      16.500000  Upper middle income
150       Sudan         SDN     33.477      22.700000  Lower middle income
```

[30]:
```
df['CountryName']        # Select one column as a Series (1D)
```

[30]:
```
0                    Aruba
1              Afghanistan
2                   Angola
3                  Albania
4      United Arab Emirates
               …
190             Yemen, Rep.
191            South Africa
192       Congo, Dem. Rep.
193                  Zambia
194                Zimbabwe
Name: CountryName, Length: 195, dtype: object
```

[31]:
```
df[['CountryName','CountryCode','BirthRate']]   # Select multiple columns as a␣
↪DataFrame (2D)
```

[31]:
```
              CountryName CountryCode  BirthRate
0                   Aruba         ABW     10.244
1             Afghanistan         AFG     35.253
2                  Angola         AGO     45.985
3                 Albania         ALB     12.877
4    United Arab Emirates         ARE     11.044
..                    …         …       …
190           Yemen, Rep.         YEM     32.947
191          South Africa         ZAF     20.850
192     Congo, Dem. Rep.          COD     42.394
193                Zambia         ZMB     40.471
194              Zimbabwe         ZWE     35.715

[195 rows x 3 columns]
```

More important slicing tips (must know)

[32]:
```
df.loc[5]       # Get row with index label 5 (label-based)
```

[32]:
```
CountryName        Argentina
CountryCode              ARG
BirthRate             17.716
InternetUsers           59.9
IncomeGroup      High income
Name: 5, dtype: object
```

[33]:
```
df.iloc[5]      # Get the 6th row (position-based)
```

[33]:
```
CountryName        Argentina
CountryCode              ARG
BirthRate             17.716
InternetUsers           59.9
IncomeGroup      High income
Name: 5, dtype: object
```

[34]:
```
df.loc[5:10]    # Rows with index labels from 5 to 10 (inclusive)
```

[34]:
```
             CountryName CountryCode  BirthRate  InternetUsers  \
5              Argentina         ARG     17.716        59.9000
6                Armenia         ARM     13.308        41.9000
7    Antigua and Barbuda         ATG     16.447        63.4000
8              Australia         AUS     13.200        83.0000
9                Austria         AUT      9.400        80.6188
10           Azerbaijan          AZE     18.300        58.7000

             IncomeGroup
5           High income
6   Lower middle income
7           High income
8           High income
9           High income
10  Upper middle income
```

[35]:
```
df.iloc[5:10]   # Rows 5 to 9 (exclusive of 10, like normal Python slicing)
```

[35]:
```
            CountryName CountryCode  BirthRate  InternetUsers  \
5             Argentina         ARG     17.716        59.9000
6               Armenia         ARM     13.308        41.9000
7   Antigua and Barbuda         ATG     16.447        63.4000
8             Australia         AUS     13.200        83.0000
9               Austria         AUT      9.400        80.6188

            IncomeGroup
5          High income
6  Lower middle income
7          High income
8          High income
```

```
9          High income
```

[36]:
```
df[['CountryName']]      # Returns DataFrame (not Series)
```

[36]:
```
              CountryName
0                   Aruba
1             Afghanistan
2                  Angola
3                 Albania
4    United Arab Emirates
..                    …
190           Yemen, Rep.
191          South Africa
192      Congo, Dem. Rep.
193                Zambia
194              Zimbabwe

[195 rows x 1 columns]
```

[37]:
```
df['CountryName'][0:5]    # First 5 values from the 'CountryName' column
```

[37]:
```
0                   Aruba
1             Afghanistan
2                  Angola
3                 Albania
4    United Arab Emirates
Name: CountryName, dtype: object
```

3 We created a new DataFrame `df_cat` that includes only the categorical columns: `CountryName`, `CountryCode`, and `IncomeGroup`

[38]:
```
df.columns
```

[38]:
```
Index(['CountryName', 'CountryCode', 'BirthRate', 'InternetUsers',
       'IncomeGroup'],
      dtype='object')
```

[39]:
```
df_cat = df[['CountryName', 'CountryCode',
       'IncomeGroup']]
```

[40]:
```
df_cat
```

[40]:
```
              CountryName CountryCode          IncomeGroup
0                   Aruba         ABW          High income
1             Afghanistan         AFG           Low income
2                  Angola         AGO  Upper middle income
3                 Albania         ALB  Upper middle income
```

```
4    United Arab Emirates         ARE          High income
..                    …        …                    …
190           Yemen, Rep.         YEM  Lower middle income
191          South Africa         ZAF  Upper middle income
192      Congo, Dem. Rep.         COD           Low income
193                Zambia         ZMB  Lower middle income
194              Zimbabwe         ZWE           Low income

[195 rows x 3 columns]
```