# project 3

# country gdp analysis using pandas and seaborn

In [49]:
```
import pandas as pd
```

In [50]:
```
pd.__version__
```

Out[50]:
```
'2.3.0'
```

# pd.read_ functions in panda

| Function | Description |
|---|---|
| `pd.read_csv()` | Read a CSV file |
| `pd.read_excel()` | Read an Excel file |
| `pd.read_json()` | Read a JSON file |
| `pd.read_sql()` | Read from a SQL database |
| `pd.read_html()` | Read HTML tables |
| `pd.read_parquet()` | Read a Parquet file |
| `pd.read_pickle()` | Load a pickled pandas object |
| `pd.read_table()` | Read a general delimited file (default is tab `\t`) |
| `pd.read_feather()` | Read Feather-format file |
| `pd.read_sas()` | Read SAS datasets |
| `pd.read_stata()` | Read Stata files |
| `pd.read_clipboard()` | Read data copied to the clipboard (like a table from Excel or browser) |

# 📋 Basic Attributes of a DataFrame

| Attribute | Description |
|---|---|
| `df.columns` | Returns an Index object containing column names |
| `df.index` | Returns the row labels (index) of the DataFrame |

| Attribute | Description |
|---|---|
| `df.dtypes` | Shows the data types of each column |
| `df.shape` | Tuple of (rows, columns) |
| `df.size` | Total number of elements (rows × columns) |
| `df.ndim` | Number of dimensions (2 for DataFrame) |
| `df.values` | Numpy array representation of the DataFrame (avoid for large data) |
| `df.T` | Transpose of the DataFrame (rows ↔ columns) |
| `df.axes` | List of the row and column axis labels |
| `df.empty` | Returns `True` if DataFrame is empty |
| `df.memory_usage()` | Memory used by each column (in bytes) |
| `df.attrs` | Dictionary to store custom metadata (user-defined) |
| `df.style` | Returns a Styler object to apply formatting for display |
| `df.select_dtypes()` | Select columns by data type |

# 📊 Summary / Info

| Attribute/Method | Description |
|---|---|
| `df.info()` | Summary of DataFrame (columns, non-null counts, types) |
| `df.describe()` | Summary statistics (for numeric columns) |
| `df.head(n)` | First `n` rows (default 5) |
| `df.tail(n)` | Last `n` rows (default 5) |
| `df.sample(n)` | Random sample of `n` rows |

# 🔍 Data Inspection

| Method / Attribute | Description | Example | C Exp |
|---|---|---|---|
| `df.isnull()` | Returns True for each cell that is missing ( `NaN` ) | `df.isnull()` | DataFr `True` / helps l missing |

| Method / Attribute | Description | Example | O Exp |
|---|---|---|---|
| `df.notnull()` | Opposite of `isnull()`. True if value is present | `df.notnull()` | DataFra `True` / identifi missing |
| `df.isnull().sum()` | Counts missing values in each column | `df.isnull().sum()` | Series many each co |
| `df.count()` | Counts non-null (non-missing) values per column | `df.count()` | Useful unders comple columr |
| `df.duplicated()` | Returns `True` for each duplicated row | `df.duplicated()` | Helps c remove rows |
| `df.drop_duplicates()` | Returns a DataFrame with duplicate rows removed | `df.drop_duplicates()` | Cleane DataFra remain unless |
| `df.nunique()` | Counts the number of unique values in each column | `df.nunique()` | Helps u catego unique |
| `df['col'].unique()` | Lists all unique values in a specific column | `df['Gender'].unique()` | Shows values `['Mal 'Femal` |
| `df['col'].value_counts()` | Shows how often each unique value appears in a column | `df['Gender'].value_counts()` | Freque like: `M Female` |
| `df.corr()` | Calculates correlation between numeric columns | `df.corr()` | Correla 1 mear positive negativ correla |
| `df.memory_usage()` | Shows how much memory each column consumes (in bytes) | `df.memory_usage()` | Useful optimiz perforn |

In [3]:
```
# dir(df) # to see everything pandas offers
```

In [52]:
```
df=pd.read_csv(r"C:\Users\User\Downloads\assingments\Projects\p3\data set\da
df
```

Out[52]:

| | 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 |
| **3** | Albania | ALB | 12.877 | 57.2 | Upper middle income |
| **4** | United Arab Emirates | ARE | 11.044 | 88.0 | High income |
| **...** | ... | ... | ... | ... | ... |
| **190** | Yemen, Rep. | YEM | 32.947 | 20.0 | Lower middle income |
| **191** | South Africa | ZAF | 20.850 | 46.5 | Upper middle income |
| **192** | Congo, Dem. Rep. | COD | 42.394 | 2.2 | Low income |
| **193** | Zambia | ZMB | 40.471 | 15.4 | Lower middle income |
| **194** | Zimbabwe | ZWE | 35.715 | 18.5 | Low income |

195 rows × 5 columns

In [53]:
```
print(id(df))
print(type(df))
```

```
2526630135760
<class 'pandas.core.frame.DataFrame'>
```

In [54]:
```
df.columns
```

Out[54]:
```
Index(['CountryName', 'CountryCode', 'BirthRate', 'InternetUsers',
       'IncomeGroup'],
      dtype='object')
```

In [55]:
```
len(df.columns)
```

Out[55]: 5

In [56]:
```
df.shape
```

Out[56]: (195, 5)

In [57]:
```
df.isnull() #false= no missing values , True =  missing values
```

Out[57]:

| | 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 × 5 columns

In [58]:
```
df.isna()
```

Out[58]:

| | 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 × 5 columns

In [59]:
```
df.isnull().sum()
```

Out[59]:
```
CountryName      0
CountryCode      0
BirthRate        0
InternetUsers    0
IncomeGroup      0
dtype: int64
```

In [60]:
```
df.head()
```

Out[60]:

| | 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 |
| **3** | Albania | ALB | 12.877 | 57.2 | Upper middle income |
| **4** | United Arab Emirates | ARE | 11.044 | 88.0 | High income |

In [61]:
```
df.tail()
```

Out[61]:

| | CountryName | CountryCode | BirthRate | InternetUsers | IncomeGroup |
|---|---|---|---|---|---|
| **190** | Yemen, Rep. | YEM | 32.947 | 20.0 | Lower middle income |
| **191** | South Africa | ZAF | 20.850 | 46.5 | Upper middle income |
| **192** | Congo, Dem. Rep. | COD | 42.394 | 2.2 | Low income |
| **193** | Zambia | ZMB | 40.471 | 15.4 | Lower middle income |
| **194** | Zimbabwe | ZWE | 35.715 | 18.5 | Low income |

# 📘 Common dtypes in pandas

| dtype | Meaning |
|---|---|
| `object` | Text/string data |
| `int64` | Integer numbers |
| `float64` | Decimal numbers |
| `bool` | Boolean values ( `True` / `False` ) |
| `datetime64` | Date and time values |
| `category` | Categorical data (optimized storage) |

# 📌 Less common but useful

| Attribute | Description |
| --- | --- |
| `df.attrs` | Custom metadata (like a notes dictionary) |
| `df.flags` | Info about the underlying ndarray flags |
| `df.index.name` | Name of the index |
| `df.columns.name` | Name of the column index |
| `df._data` | Internal 2D block manager (advanced use) |

In [62]:
```
df.dtypes
```

Out[62]:
```
CountryName       object
CountryCode       object
BirthRate        float64
InternetUsers    float64
IncomeGroup       object
dtype: object
```

In [63]:
```
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
```

In [64]:
```
print(df.memory_usage())  #Memory used by each column (in bytes)
```

```
Index             132
CountryName      1560
CountryCode      1560
BirthRate        1560
InternetUsers    1560
IncomeGroup      1560
dtype: int64
```

In [65]:
```
print("1560 + 1560 + 1560 + 1560 + 1560 = ",1560 + 1560 + 1560 + 1560 + 156
```

```
1560 + 1560 + 1560 + 1560 + 1560 =  7800
```

# 🔪 Slicing in Python and Pandas

# 🧭 Using .loc[] – Label-based

`df.loc[row_start:row_end, col_start:col_end]`

| Feature | Works with labels (names) | End inclusive |
| --- | --- | --- |
| Rows | Yes | ✅ Yes |
| Columns | Yes | ✅ Yes |

# 🧮 Using .iloc[] – Integer-based

`df.iloc[ row_start: row_end , col_start : col_end]`

| Feature | Works with positions (0-based) | End exclusive |
| --- | --- | --- |
| Rows | Yes | ❌ No |
| Columns | Yes | ❌ No |

📘 Example: `df.iloc[0:3, 1:3]`

# ✅ Shortcut: Slice All Rows or Columns

`df[:]` # All rows
`df[:, :]*` # All rows and all columns (if using NumPy)

| Feature | `.loc[]` | `.iloc[]` |
| --- | --- | --- |
| Indexing type | Label-based | Position-based |
| End index | Inclusive | Exclusive |
| Supports slice | Yes | Yes |
| Example | `df.loc['A':'C']` | `df.iloc[0:3]` |

In [66]:
```
df[:]
```

Out[66]:

| | 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 |
| **3** | Albania | ALB | 12.877 | 57.2 | Upper middle income |
| **4** | United Arab Emirates | ARE | 11.044 | 88.0 | High income |
| **...** | ... | ... | ... | ... | ... |
| **190** | Yemen, Rep. | YEM | 32.947 | 20.0 | Lower middle income |
| **191** | South Africa | ZAF | 20.850 | 46.5 | Upper middle income |
| **192** | Congo, Dem. Rep. | COD | 42.394 | 2.2 | Low income |
| **193** | Zambia | ZMB | 40.471 | 15.4 | Lower middle income |
| **194** | Zimbabwe | ZWE | 35.715 | 18.5 | Low income |

195 rows × 5 columns

In [67]:

```
df[::-1] #reverse
```

Out[67]:

| | CountryName | CountryCode | BirthRate | InternetUsers | IncomeGroup |
|---|---|---|---|---|---|
| **194** | Zimbabwe | ZWE | 35.715 | 18.5 | Low income |
| **193** | Zambia | ZMB | 40.471 | 15.4 | Lower middle income |
| **192** | Congo, Dem. Rep. | COD | 42.394 | 2.2 | Low income |
| **191** | South Africa | ZAF | 20.850 | 46.5 | Upper middle income |
| **190** | Yemen, Rep. | YEM | 32.947 | 20.0 | Lower middle income |
| **...** | ... | ... | ... | ... | ... |
| **4** | United Arab Emirates | ARE | 11.044 | 88.0 | High income |
| **3** | Albania | ALB | 12.877 | 57.2 | Upper middle income |
| **2** | Angola | AGO | 45.985 | 19.1 | Upper middle income |
| **1** | Afghanistan | AFG | 35.253 | 5.9 | Low income |
| **0** | Aruba | ABW | 10.244 | 78.9 | High income |

195 rows × 5 columns

In [68]:

```
df[:11] # return row fron 0 To 10 index
```

Out[68]:

| | CountryName | CountryCode | BirthRate | InternetUsers | IncomeGroup |
|---|---|---|---|---|---|
| **0** | Aruba | ABW | 10.244 | 78.9000 | High income |
| **1** | Afghanistan | AFG | 35.253 | 5.9000 | Low income |
| **2** | Angola | AGO | 45.985 | 19.1000 | Upper middle income |
| **3** | Albania | ALB | 12.877 | 57.2000 | Upper middle income |
| **4** | United Arab Emirates | ARE | 11.044 | 88.0000 | High income |
| **5** | Argentina | ARG | 17.716 | 59.9000 | High income |
| **6** | Armenia | ARM | 13.308 | 41.9000 | Lower middle income |
| **7** | Antigua and Barbuda | ATG | 16.447 | 63.4000 | High income |
| **8** | Australia | AUS | 13.200 | 83.0000 | High income |
| **9** | Austria | AUT | 9.400 | 80.6188 | High income |
| **10** | Azerbaijan | AZE | 18.300 | 58.7000 | Upper middle income |

In [69]:

```
df[0:200:50]
```

Out[69]:

| | 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 |

In [70]:

```
df
```

Out[70]:

| | 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 |
| **3** | Albania | ALB | 12.877 | 57.2 | Upper middle income |
| **4** | United Arab Emirates | ARE | 11.044 | 88.0 | High income |
| **...** | ... | ... | ... | ... | ... |
| **190** | Yemen, Rep. | YEM | 32.947 | 20.0 | Lower middle income |
| **191** | South Africa | ZAF | 20.850 | 46.5 | Upper middle income |
| **192** | Congo, Dem. Rep. | COD | 42.394 | 2.2 | Low income |
| **193** | Zambia | ZMB | 40.471 | 15.4 | Lower middle income |
| **194** | Zimbabwe | ZWE | 35.715 | 18.5 | Low income |

195 rows × 5 columns

# ⚠️ print a specific coulmn:

You must use $\boxed{\text{double square brackets\_}}$ [[...]] to select multiple columns.

A single column: `df["CountryName"]` returns a Series.

Multiple columns: `df[["CountryName", "CountryCode"]]` returns a DataFrame.

In [71]:
```
df["CountryName"]
```

Out[71]:
```
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
```

In [72]:
```
df[ ["CountryName","CountryCode"] ] #double square brackets_ [[...]] to sele
```

Out[72]:

| | CountryName | CountryCode |
|---|---|---|
| **0** | Aruba | ABW |
| **1** | Afghanistan | AFG |
| **2** | Angola | AGO |
| **3** | Albania | ALB |
| **4** | United Arab Emirates | ARE |
| **...** | ... | ... |
| **190** | Yemen, Rep. | YEM |
| **191** | South Africa | ZAF |
| **192** | Congo, Dem. Rep. | COD |
| **193** | Zambia | ZMB |
| **194** | Zimbabwe | ZWE |

195 rows × 2 columns

In [73]:
```
df[ ["CountryName","CountryCode",'BirthRate'] ]
```

Out[73]:

| | 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 × 3 columns

In [74]:
```
df.head(3) # if the valuesis mentionefd, then the first n rows will br retur
```

Out[74]:

| | 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 |

# descriptive Statistics

## $df.describe()$ :- *describe fuction will only print numerical numberical columns*

## 📊 Numeric Data

| Data Type | Behavior | Returned Stats | Detailed Description |
|---|---|---|---|
| **Numeric** | Summarizes statistical metrics | `count`, `mean`, `std`, `min`, `25%`, `50%`, `75%`, `max` | Provides statistical summary of numerical columns. Shows central tendency, spread, and distribution of values.<br><br>**Details:**<br>• `count`: Number of non-null values<br>• `mean`: Average value<br>• `std`: Standard deviation (spread)<br>• `min`: Minimum value<br>• `25%`: First quartile (Q1)<br>• `50%`: Median (Q2)<br>• `75%`: Third quartile (Q3)<br>• `max`: Maximum value |

## 🔤 Categorical Data

| Data Type | Behavior | Returned Stats | Detailed Description |
|---|---|---|---|
| **Categorical** | Summarizes frequency-based metrics | `count`, `unique`, `top`, `freq` | Gives insights into text/object data. Shows most frequent category and distribution of unique values. |

| Data Type | Behavior | Returned Stats | Detailed Description |
| --- | --- | --- | --- |
| | | | **Details:**<br>• `count` : Number of non-null values<br>• `unique` : Number of distinct entries<br>• `top` : Most frequent value (mode)<br>• `freq` : Frequency of the top value |

In [75]:
```
df.describe()
```

Out[75]:

| | 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 |

In [76]:
```
df.columns
```

Out[76]:
```
Index(['CountryName', 'CountryCode', 'BirthRate', 'InternetUsers',
       'IncomeGroup'],
      dtype='object')
```

In [77]:
```
df_cat=df[ ["CountryName","CountryCode",'IncomeGroup'] ]
df_cat
```

Out[77]:

| | 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 × 3 columns

In [78]:

```
df_cat.describe()
```

Out[78]:

| | CountryName | CountryCode | IncomeGroup |
|---|---|---|---|
| **count** | 195 | 195 | 195 |
| **unique** | 195 | 195 | 4 |
| **top** | Aruba | ABW | High income |
| **freq** | 1 | 1 | 67 |

In [79]:

```
df.head(2)
```

Out[79]:

| | CountryName | CountryCode | BirthRate | InternetUsers | IncomeGroup |
|---|---|---|---|---|---|
| **0** | Aruba | ABW | 10.244 | 78.9 | High income |
| **1** | Afghanistan | AFG | 35.253 | 5.9 | Low income |

In [80]:

```
df.columns
```

Out[80]:

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

# Renameing columns

In [81]:

```
df.columns = ["A","B","C","D","E"]  #RENameing columns
```

In [82]:
```
df
```

Out[82]:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **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 |
| **3** | Albania | ALB | 12.877 | 57.2 | Upper middle income |
| **4** | United Arab Emirates | ARE | 11.044 | 88.0 | High income |
| **...** | ... | ... | ... | ... | ... |
| **190** | Yemen, Rep. | YEM | 32.947 | 20.0 | Lower middle income |
| **191** | South Africa | ZAF | 20.850 | 46.5 | Upper middle income |
| **192** | Congo, Dem. Rep. | COD | 42.394 | 2.2 | Low income |
| **193** | Zambia | ZMB | 40.471 | 15.4 | Lower middle income |
| **194** | Zimbabwe | ZWE | 35.715 | 18.5 | Low income |

195 rows × 5 columns

In [83]:
```
df.columns = [ "CountryName",   "CountryCode",  "BirthRate",    "InternetUse
```

In [84]:
```
df
```

Out[84]:

| | 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 |
| **3** | Albania | ALB | 12.877 | 57.2 | Upper middle income |
| **4** | United Arab Emirates | ARE | 11.044 | 88.0 | High income |
| **...** | ... | ... | ... | ... | ... |
| **190** | Yemen, Rep. | YEM | 32.947 | 20.0 | Lower middle income |
| **191** | South Africa | ZAF | 20.850 | 46.5 | Upper middle income |
| **192** | Congo, Dem. Rep. | COD | 42.394 | 2.2 | Low income |
| **193** | Zambia | ZMB | 40.471 | 15.4 | Lower middle income |
| **194** | Zimbabwe | ZWE | 35.715 | 18.5 | Low income |

195 rows × 5 columns

In [85]:

```
df_categorial = df[["CountryName","CountryCode","IncomeGroup"]]
df_categorial.head()
```

Out[85]:

| | 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 |

In [86]:

```
df_categorial.describe()
```

Out[86]:

| | CountryName | CountryCode | IncomeGroup |
|---|---|---|---|
| **count** | 195 | 195 | 195 |
| **unique** | 195 | 195 | 4 |
| **top** | Aruba | ABW | High income |
| **freq** | 1 | 1 | 67 |

In [87]:
```
["CountryName","BirthRate"]
```

Out[87]:
```
['CountryName', 'BirthRate']
```

In [88]:
```
df[["CountryName","BirthRate"]]
```

Out[88]:

| | CountryName | BirthRate |
|---|---|---|
| 0 | Aruba | 10.244 |
| 1 | Afghanistan | 35.253 |
| 2 | Angola | 45.985 |
| 3 | Albania | 12.877 |
| 4 | United Arab Emirates | 11.044 |
| ... | ... | ... |
| 190 | Yemen, Rep. | 32.947 |
| 191 | South Africa | 20.850 |
| 192 | Congo, Dem. Rep. | 42.394 |
| 193 | Zambia | 40.471 |
| 194 | Zimbabwe | 35.715 |

195 rows × 2 columns

In [89]:
```
df[["CountryName","BirthRate","IncomeGroup"]]
```

Out[89]:

| | CountryName | BirthRate | IncomeGroup |
|---|---|---|---|
| 0 | Aruba | 10.244 | High income |
| 1 | Afghanistan | 35.253 | Low income |
| 2 | Angola | 45.985 | Upper middle income |
| 3 | Albania | 12.877 | Upper middle income |
| 4 | United Arab Emirates | 11.044 | High income |
| ... | ... | ... | ... |
| 190 | Yemen, Rep. | 32.947 | Lower middle income |
| 191 | South Africa | 20.850 | Upper middle income |
| 192 | Congo, Dem. Rep. | 42.394 | Low income |
| 193 | Zambia | 40.471 | Lower middle income |
| 194 | Zimbabwe | 35.715 | Low income |

195 rows × 3 columns

In [90]:
```
df.head()
```

Out[90]:

| | 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 |
| **3** | Albania | ALB | 12.877 | 57.2 | Upper middle income |
| **4** | United Arab Emirates | ARE | 11.044 | 88.0 | High income |

In [91]:
```
df.BirthRate*df.InternetUsers
```

Out[91]:
```
0      808.2516
1      207.9927
2      878.3135
3      736.5644
4      971.8720
         ...
190    658.9400
191    969.5250
192     93.2668
193    623.2534
194    660.7275
Length: 195, dtype: float64
```

In [92]:
```
# df[[BirthRate * InternetUsers]]
# NameError: name 'BirthRate' is not defined
```

In [93]:
```
df["myCalc"] = df.BirthRate*df.InternetUsers
```

In [94]:
```
df
```

Out[94]:

| | CountryName | CountryCode | BirthRate | InternetUsers | IncomeGroup | m |
|---|---|---|---|---|---|---|
| **0** | Aruba | ABW | 10.244 | 78.9 | High income | 808 |
| **1** | Afghanistan | AFG | 35.253 | 5.9 | Low income | 207 |
| **2** | Angola | AGO | 45.985 | 19.1 | Upper middle income | 878 |
| **3** | Albania | ALB | 12.877 | 57.2 | Upper middle income | 736 |
| **4** | United Arab Emirates | ARE | 11.044 | 88.0 | High income | 971 |
| **...** | ... | ... | ... | ... | ... | |
| **190** | Yemen, Rep. | YEM | 32.947 | 20.0 | Lower middle income | 658 |
| **191** | South Africa | ZAF | 20.850 | 46.5 | Upper middle income | 969 |
| **192** | Congo, Dem. Rep. | COD | 42.394 | 2.2 | Low income | 93 |
| **193** | Zambia | ZMB | 40.471 | 15.4 | Lower middle income | 623 |
| **194** | Zimbabwe | ZWE | 35.715 | 18.5 | Low income | 660 |

195 rows × 6 columns

In [95]:

```
df.head()
```

Out[95]:

| | CountryName | CountryCode | BirthRate | InternetUsers | IncomeGroup | myC |
|---|---|---|---|---|---|---|
| **0** | Aruba | ABW | 10.244 | 78.9 | High income | 808.2 |
| **1** | Afghanistan | AFG | 35.253 | 5.9 | Low income | 207.9 |
| **2** | Angola | AGO | 45.985 | 19.1 | Upper middle income | 878.3 |
| **3** | Albania | ALB | 12.877 | 57.2 | Upper middle income | 736.5 |
| **4** | United Arab Emirates | ARE | 11.044 | 88.0 | High income | 971.8 |

# ✅ df.drop() in pandas — Full Explanation

The `drop()` method is used to remove rows or columns from a pandas `DataFrame`.

🔹 General Syntax
`df.drop(labels, axis=0, inplace=False)`

| Parameter | Description |
|---|---|
| `labels` | Name(s) or index(es) to drop |
| `axis` | `0` for **rows**, `1` for **columns** |
| `inplace` | `True` to modify the original DataFrame |
| `errors` | `'ignore'` to skip labels that don't exist |

# 🔥 Pro Tip

Use errors='ignore'# to avoid crashes:
`df.drop('NonExistentColumn', axis=1, errors='ignore')`

In [96]:
```
df.drop("myCalc",axis=1)
```

Out[96]:

| | 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 |
| **3** | Albania | ALB | 12.877 | 57.2 | Upper middle income |
| **4** | United Arab Emirates | ARE | 11.044 | 88.0 | High income |
| **...** | ... | ... | ... | ... | ... |
| **190** | Yemen, Rep. | YEM | 32.947 | 20.0 | Lower middle income |
| **191** | South Africa | ZAF | 20.850 | 46.5 | Upper middle income |
| **192** | Congo, Dem. Rep. | COD | 42.394 | 2.2 | Low income |
| **193** | Zambia | ZMB | 40.471 | 15.4 | Lower middle income |
| **194** | Zimbabwe | ZWE | 35.715 | 18.5 | Low income |

195 rows × 5 columns

In [97]:
```
df.columns
```

Out[97]:
```
Index(['CountryName', 'CountryCode', 'BirthRate', 'InternetUsers',
       'IncomeGroup', 'myCalc'],
      dtype='object')
```

In [98]:
```
df
```

Out[98]:

| | CountryName | CountryCode | BirthRate | InternetUsers | IncomeGroup | m |
|---|---|---|---|---|---|---|
| **0** | Aruba | ABW | 10.244 | 78.9 | High income | 808 |
| **1** | Afghanistan | AFG | 35.253 | 5.9 | Low income | 207 |
| **2** | Angola | AGO | 45.985 | 19.1 | Upper middle income | 878 |
| **3** | Albania | ALB | 12.877 | 57.2 | Upper middle income | 736 |
| **4** | United Arab Emirates | ARE | 11.044 | 88.0 | High income | 971 |
| **...** | ... | ... | ... | ... | ... | |
| **190** | Yemen, Rep. | YEM | 32.947 | 20.0 | Lower middle income | 658 |
| **191** | South Africa | ZAF | 20.850 | 46.5 | Upper middle income | 969 |
| **192** | Congo, Dem. Rep. | COD | 42.394 | 2.2 | Low income | 93 |
| **193** | Zambia | ZMB | 40.471 | 15.4 | Lower middle income | 623 |
| **194** | Zimbabwe | ZWE | 35.715 | 18.5 | Low income | 660 |

195 rows × 6 columns

In [99]:
```
df=df.drop("myCalc",axis=1)
df
```

Out[99]:

| | 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 |
| **3** | Albania | ALB | 12.877 | 57.2 | Upper middle income |
| **4** | United Arab Emirates | ARE | 11.044 | 88.0 | High income |
| **...** | ... | ... | ... | ... | ... |
| **190** | Yemen, Rep. | YEM | 32.947 | 20.0 | Lower middle income |
| **191** | South Africa | ZAF | 20.850 | 46.5 | Upper middle income |
| **192** | Congo, Dem. Rep. | COD | 42.394 | 2.2 | Low income |
| **193** | Zambia | ZMB | 40.471 | 15.4 | Lower middle income |
| **194** | Zimbabwe | ZWE | 35.715 | 18.5 | Low income |

195 rows × 5 columns

In [100…

```
df["InternetUsers"]
```

Out[100…

```
0      78.9
1       5.9
2      19.1
3      57.2
4      88.0
       ...
190    20.0
191    46.5
192     2.2
193    15.4
194    18.5
Name: InternetUsers, Length: 195, dtype: float64
```

In [101…

```
df[["InternetUsers"]]
```

Out[101…

| | InternetUsers |
|---|---|
| **0** | 78.9 |
| **1** | 5.9 |
| **2** | 19.1 |
| **3** | 57.2 |
| **4** | 88.0 |
| **...** | ... |
| **190** | 20.0 |
| **191** | 46.5 |
| **192** | 2.2 |
| **193** | 15.4 |
| **194** | 18.5 |

195 rows × 1 columns

In [102…

```
df.InternetUsers<2
```

Out[102…

```
0      False
1      False
2      False
3      False
4      False
       ...
190    False
191    False
192    False
193    False
194    False
Name: InternetUsers, Length: 195, dtype: bool
```

In [103…

```
filter=df.InternetUsers<2
filter
```

Out[103…

```
0      False
1      False
2      False
3      False
4      False
       ...
190    False
191    False
192    False
193    False
194    False
Name: InternetUsers, Length: 195, dtype: bool
```

In [104…

```
df[filter] # it return the values where InternetUsers is less 2
```

Out[104…

| | CountryName | CountryCode | BirthRate | InternetUsers | IncomeGroup |
|---|---|---|---|---|---|
| **11** | Burundi | BDI | 44.151 | 1.3 | Low income |
| **52** | Eritrea | ERI | 34.800 | 0.9 | Low income |
| **55** | Ethiopia | ETH | 32.925 | 1.9 | Low income |
| **64** | Guinea | GIN | 37.337 | 1.6 | Low income |
| **117** | Myanmar | MMR | 18.119 | 1.6 | Lower middle income |
| **127** | Niger | NER | 49.661 | 1.7 | Low income |
| **154** | Sierra Leone | SLE | 36.729 | 1.7 | Low income |
| **156** | Somalia | SOM | 43.891 | 1.5 | Low income |
| **172** | Timor-Leste | TLS | 35.755 | 1.1 | Lower middle income |

# Operator in data frame (df)

In [105…

```
f2=df.BirthRate  > 40
f2
```

Out[105…

```
0      False
1      False
2       True
3      False
4      False
       ...
190    False
191    False
192     True
193     True
194    False
Name: BirthRate, Length: 195, dtype: bool
```

In [106…

```
df[f2]
```

Out[106…

| | CountryName | CountryCode | BirthRate | InternetUsers | IncomeGroup |
|---|---|---|---|---|---|
| **2** | Angola | AGO | 45.985 | 19.1 | Upper middle income |
| **11** | Burundi | BDI | 44.151 | 1.3 | Low income |
| **14** | Burkina Faso | BFA | 40.551 | 9.1 | Low income |
| **65** | Gambia, The | GMB | 42.525 | 14.0 | Low income |
| **115** | Mali | MLI | 44.138 | 3.5 | Low income |
| **127** | Niger | NER | 49.661 | 1.7 | Low income |
| **128** | Nigeria | NGA | 40.045 | 38.0 | Lower middle income |
| **156** | Somalia | SOM | 43.891 | 1.5 | Low income |
| **167** | Chad | TCD | 45.745 | 2.3 | Low income |
| **178** | Uganda | UGA | 43.474 | 16.2 | Low income |
| **192** | Congo, Dem. Rep. | COD | 42.394 | 2.2 | Low income |
| **193** | Zambia | ZMB | 40.471 | 15.4 | Lower middle income |

In [107…

```
len(df[f2])
```

Out[107…

```
12
```

In [108…

```
filter & f2
```

Out[108…

```
0      False
1      False
2      False
3      False
4      False
       ...
190    False
191    False
192    False
193    False
194    False
Length: 195, dtype: bool
```

In [109…

```
df[filter & f2]
```

Out[109…

| | CountryName | CountryCode | BirthRate | InternetUsers | IncomeGroup |
|---|---|---|---|---|---|
| **11** | Burundi | BDI | 44.151 | 1.3 | Low income |
| **127** | Niger | NER | 49.661 | 1.7 | Low income |
| **156** | Somalia | SOM | 43.891 | 1.5 | Low income |

In [110…
```
df[df.IncomeGroup=="High income"]
```

Out[110…

| | CountryName | CountryCode | BirthRate | InternetUsers | IncomeGroup |
|---|---|---|---|---|---|
| **0** | Aruba | ABW | 10.244 | 78.90 | High income |
| **4** | United Arab Emirates | ARE | 11.044 | 88.00 | High income |
| **5** | Argentina | ARG | 17.716 | 59.90 | High income |
| **7** | Antigua and Barbuda | ATG | 16.447 | 63.40 | High income |
| **8** | Australia | AUS | 13.200 | 83.00 | High income |
| **...** | ... | ... | ... | ... | ... |
| **174** | Trinidad and Tobago | TTO | 14.590 | 63.80 | High income |
| **180** | Uruguay | URY | 14.374 | 57.69 | High income |
| **181** | United States | USA | 12.500 | 84.20 | High income |
| **184** | Venezuela, RB | VEN | 19.842 | 54.90 | High income |
| **185** | Virgin Islands (U.S.) | VIR | 10.700 | 45.30 | High income |

67 rows × 5 columns

In [111…
```
df[df.IncomeGroup=="Low income"]
```

Out[111…

| | CountryName | CountryCode | BirthRate | InternetUsers | IncomeGroup |
|---|---|---|---|---|---|
| **1** | Afghanistan | AFG | 35.253 | 5.90 | Low income |
| **11** | Burundi | BDI | 44.151 | 1.30 | Low income |
| **13** | Benin | BEN | 36.440 | 4.90 | Low income |
| **14** | Burkina Faso | BFA | 40.551 | 9.10 | Low income |
| **29** | Central African Republic | CAF | 34.076 | 3.50 | Low income |
| **38** | Comoros | COM | 34.326 | 6.50 | Low income |
| **52** | Eritrea | ERI | 34.800 | 0.90 | Low income |
| **55** | Ethiopia | ETH | 32.925 | 1.90 | Low income |
| **64** | Guinea | GIN | 37.337 | 1.60 | Low income |
| **65** | Gambia, The | GMB | 42.525 | 14.00 | Low income |
| **66** | Guinea-Bissau | GNB | 37.503 | 3.10 | Low income |
| **77** | Haiti | HTI | 25.345 | 10.60 | Low income |
| **93** | Cambodia | KHM | 24.462 | 6.80 | Low income |
| **99** | Liberia | LBR | 35.521 | 3.20 | Low income |
| **111** | Madagascar | MDG | 34.686 | 3.00 | Low income |
| **115** | Mali | MLI | 44.138 | 3.50 | Low income |
| **120** | Mozambique | MOZ | 39.705 | 5.40 | Low income |
| **123** | Malawi | MWI | 39.459 | 5.05 | Low income |
| **127** | Niger | NER | 49.661 | 1.70 | Low income |
| **132** | Nepal | NPL | 20.923 | 13.30 | Low income |
| **148** | Rwanda | RWA | 32.689 | 9.00 | Low income |
| **154** | Sierra Leone | SLE | 36.729 | 1.70 | Low income |
| **156** | Somalia | SOM | 43.891 | 1.50 | Low income |
| **158** | South Sudan | SSD | 37.126 | 14.10 | Low income |
| **167** | Chad | TCD | 45.745 | 2.30 | Low income |
| **168** | Togo | TGO | 36.080 | 4.50 | Low income |
| **177** | Tanzania | TZA | 39.518 | 4.40 | Low income |
| **178** | Uganda | UGA | 43.474 | 16.20 | Low income |
| **192** | Congo, Dem. Rep. | COD | 42.394 | 2.20 | Low income |
| **194** | Zimbabwe | ZWE | 35.715 | 18.50 | Low income |

In [112…

```
df.IncomeGroup.unique() #display  all
```

Out[112… array(['High income', 'Low income', 'Upper middle income',
       'Lower middle income'], dtype=object)

In [113…
```
df.IncomeGroup.nunique() #An integer: total number of distinct categories ir
```

Out[113… 4

# 🎨 Seaborn – Python Data Visualization Library

**Seaborn** is a powerful and easy-to-use Python library built on top of **matplotlib** , designed specifically for **statistical data visualization.**

🔧 Getting Started

📦 1. Install Seaborn

`pip install seaborn`

📥 2. Import It

`import seaborn as sns`

`import matplotlib.pyplot as plt`

✅ **What Makes Seaborn Special**?

- Beautiful default styles
- Integrates well with **pandas DataFrames**
- Includes **statistical plots** (e.g., boxplots, violin plots, regressions) ✅Simplifies **multi-variable plots**

# 📊 Common Seaborn Functions

| Function | Plot Type | Use For | Example Syntax |
|---|---|---|---|
| `sns.histplot()` | Histogram | Distribution of a single variable | `sns.histplot(data=df, x='Age')` |
| `sns.kdeplot()` | KDE (smooth curve) | Estimate of the distribution (density) | `sns.kdeplot(data=df['Salary'])` |
| `sns.displot()` | Histogram or KDE | Flexible distribution plot (hist or KDE or both) | `sns.displot(data=df, x='Age', kind='kde')` |
| `sns.boxplot()` | Boxplot | Show spread, median, and outliers | `sns.boxplot(x='Gender', y='Salary', data=df)` |

| Function | Plot Type | Use For | Example Syntax |
|---|---|---|---|
| `sns.violinplot()` | Violin plot | KDE + Boxplot together (shape + stats) | `sns.violinplot(x='Gender', y='Salary', data=df)` |
| `sns.stripplot()` | Jittered dots | Raw data points (can overlap) | `sns.stripplot(x='Gender', y='Salary', data=df, jitter=True)` |
| `sns.swarmplot()` | Swarmplot (non-overlap) | Like stripplot but avoids overlapping dots | `sns.swarmplot(x='Gender', y='Salary', data=df)` |
| `sns.countplot()` | Barplot (counts) | Count of observations for each category | `sns.countplot(x='IncomeGroup', data=df)` |
| `sns.barplot()` | Barplot (summary stat) | Shows average (or other stat) + CI | `sns.barplot(x='Gender', y='Salary', data=df)` |
| `sns.pointplot()` | Line on points | Point + error bars across categories | `sns.pointplot(x='Gender', y='Score', data=df)` |
| `sns.scatterplot()` | Scatterplot | Plot relationship between two numerical variables | `sns.scatterplot(x='Age', y='Salary', data=df)` |
| `sns.lineplot()` | Lineplot | Trends over time or index | `sns.lineplot(x='Year', y='Sales', data=df)` |
| `sns.regplot()` | Regression (scatter + fit) | Scatterplot with linear regression line | `sns.regplot(x='Age', y='Salary', data=df)` |
| `sns.lmplot()` | Regression plot (grid) | Like `regplot` but with facet support | `sns.lmplot(x='Age', y='Salary', data=df, hue='Gender')` |
| `sns.heatmap()` | Heatmap | Matrix-like data (e.g. correlation) | `sns.heatmap(df.corr(), annot=True, cmap='coolwarm')` |
| `sns.pairplot()` | Pairwise plot grid | All pairwise plots with optional hue | `sns.pairplot(df, hue='Species')` |
| `sns.jointplot()` | Joint + Marginal plots | Scatterplot + histogram or KDE of each axis | `sns.jointplot(x='Age', y='Salary', data=df, kind='kde')` |

| Function | Plot Type | Use For | Example Syntax |
|---|---|---|---|
| `sns.catplot()` | Categorical plots (wrapper) | Combines `boxplot`, `violinplot`, etc. with facets | `sns.catplot(x='Gender', y='Salary', kind='box', data=df)` |
| `sns.clustermap()` | Clustered Heatmap | Hierarchical clustering heatmap | `sns.clustermap(df.corr())` |
| `sns.FacetGrid()` | Multi-plot grid | Create subplots by column/row for deeper comparison | `g = sns.FacetGrid(df, col='Gender')` `g.map(sns.histplot, 'Age')` |
| `sns.set_style()` | Style control | Set background style: `white`, `dark`, `ticks`, `whitegrid`, `darkgrid` | `sns.set_style('whitegrid')` |
| `sns.set_palette()` | Color palette | Change the color scheme | `sns.set_palette('pastel')` |

# General Formulas for Seaborn Functions (Step-by-Step Order)

# 1. 📊 Distribution Plots

Used to understand the distribution of a single variable.

| Function | General Formula |
|---|---|
| `sns.histplot()` | `sns.histplot(data=df, x='column')` |
| `sns.kdeplot()` | `sns.kdeplot(data=df['column'])` |
| `sns.displot()` | `sns.displot(data=df, x='column', kind='hist' or 'kde')` |

# 2. 🧮 Categorical Plots

Used to compare categorical groupings (e.g., gender, income group).

| Function | General Formula |
|---|---|
| `sns.countplot()` | `sns.countplot(data=df, x='category_column')` |
| `sns.barplot()` | `sns.barplot(data=df, x='category', y='value')` |
| `sns.boxplot()` | `sns.boxplot(data=df, x='category', y='value')` |
| `sns.violinplot()` | `sns.violinplot(data=df, x='category', y='value')` |
| `sns.stripplot()` | `sns.stripplot(data=df, x='category', y='value', jitter=True)` |
| `sns.swarmplot()` | `sns.swarmplot(data=df, x='category', y='value')` |

# 3. 🔗 Relationship Plots

Used to analyze relationships between numeric variables.

| Function | General Formula |
|---|---|
| `sns.scatterplot()` | `sns.scatterplot(data=df, x='var1', y='var2')` |
| `sns.lineplot()` | `sns.lineplot(data=df, x='time', y='value')` |
| `sns.regplot()` | `sns.regplot(data=df, x='var1', y='var2')` |
| `sns.lmplot()` | `sns.lmplot(data=df, x='var1', y='var2', hue='category')` |

# 4. 🔁 Multi-Variable/Matrix Plots

Used to analyze pairwise relationships or matrices.

| Function | General Formula |
|---|---|
| `sns.heatmap()` | `sns.heatmap(data=df.corr(), annot=True)` |
| `sns.clustermap()` | `sns.clustermap(data=df.corr())` |
| `sns.pairplot()` | `sns.pairplot(data=df, hue='category')` |
| `sns.jointplot()` | `sns.jointplot(data=df, x='var1', y='var2', kind='scatter')` |

# 5. 🧱 Grid / Facet Plots

Used for creating multiple plots by subgroups.

| Function | General Formula |
|---|---|
| `sns.catplot()` | `sns.catplot(data=df, x='category', y='value', kind='box')` |
| `sns.FacetGrid()` | `g = sns.FacetGrid(df, col='column');` `g.map(sns.histplot, 'x')` |

# 6. 🎨 Styling Functions

| Function | General Formula |
|---|---|
| `sns.set_style()` | `sns.set_style('whitegrid')` |
| `sns.set_palette()` | `sns.set_palette('pastel')` |

# 🌟 Bonus: Display Plots

`import matplotlib.pyplot as plt` `plt.show()`

# ✅ % matplotlib inline — What It Means

`%matplotlib` inline is a **magic** command used in **Jupyter Notebooks** to ensure that all **Matplotlib plots** are displayed **directly below the code cell** that produces them.

---

📌 Purpose:

- Renders plots **inline** (within the notebook).
- Keeps plots **visible in output cells** instead of popping up in a separate window.

⚠️ Notes:

- Only needed in **Jupyter or Colab**.
- It's a type of **IPython magic command** — specific to interactive environments.
- Not required in standard **.py** scripts (you use **plt.show()** instead).

# ✅ plt.rcParams['figure.figsize'] = (6, 2)

This line sets the default size of all future Matplotlib figures (plots) globally in your session.

## 📌 Syntax:

```python
plt.rcParams['figure.figsize'] = (width, height)
```

- `width = 6`
- `height = 2`
- Units are in **inches**

---

## 🎯 Purpose:

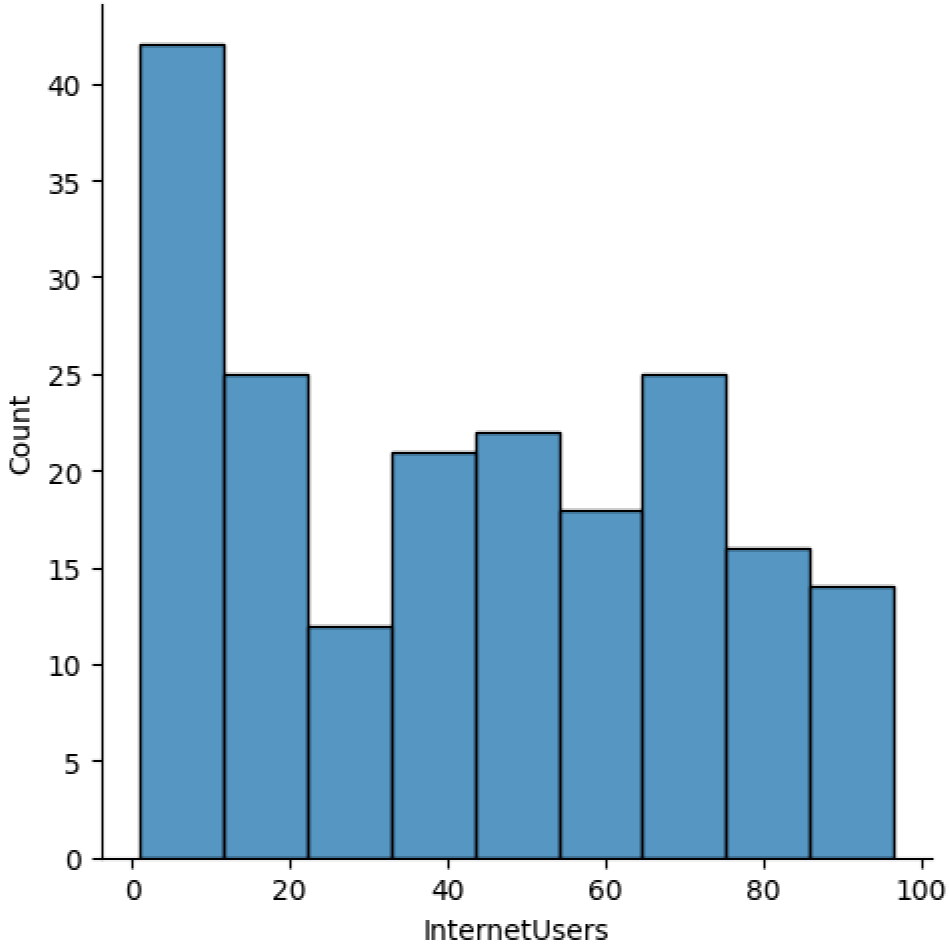
- Controls the **overall size** of plots (e.g., how wide and tall).
- Useful when you want **consistent sizing** for all plots.

In [114…
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.rcParams['figure.figsize']=6,2
import warnings
warnings.filterwarnings("ignore")
```

In [115…
```
df["InternetUsers"]
```

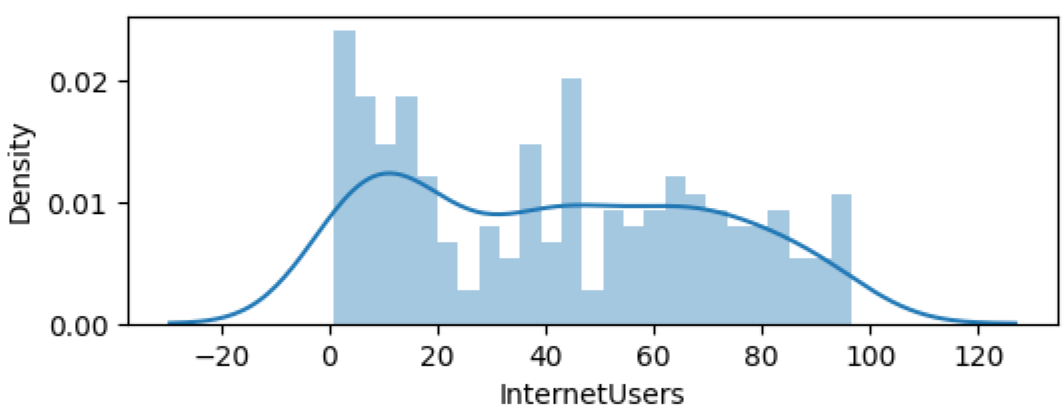
Out[115…
```
0      78.9
1       5.9
2      19.1
3      57.2
4      88.0
       ...
190    20.0
191    46.5
192     2.2
193    15.4
194    18.5
Name: InternetUsers, Length: 195, dtype: float64
```

In [116…
```
vis1= sns.displot(df["InternetUsers"])
```




In [117…

```
%matplotlib inline
plt.rcParams['figure.figsize']=6,2
vis1= sns.distplot(df["InternetUsers"],bins=25)
```




You're trying to create a Seaborn distribution plot using sns.distplot() with x, y, and hue. However:

> ⚠️ sns.distplot() is deprecated (removed in newer versions of Seaborn).

✅ Updated Alternative: Use sns.displot() or sns.histplot() for distributions If you're trying to visualize the distribution of InternetUsers grouped by IncomeGroup, here's what to do:

### ✅ Option 1: Histogram by Category (`hue`)

python Copy code

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Histogram with hue (categorical split)
vis2 = sns.histplot(data=df, x="InternetUsers", hue="IncomeGroup", kde=True)
plt.title("Distribution of Internet Users by Income Group")
plt.show()
```

### ✅ Option 2: KDE (Smooth Density Curve) by Category

python Copy code

```python
sns.kdeplot(data=df, x="InternetUsers", hue="IncomeGroup", fill=True)
plt.title("KDE of Internet Users by Income Group")
plt.show()
```

✅ Summary

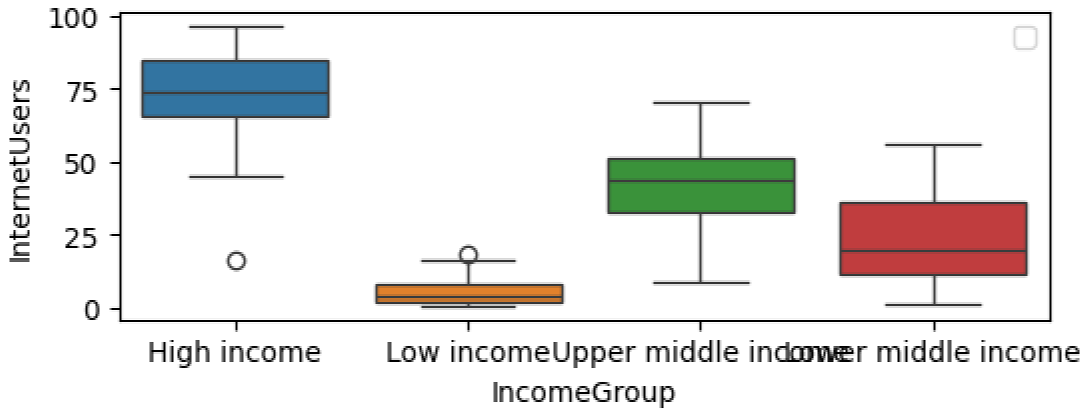
| You Want To... | Use This Function |
|---|---|
| Distribution (histogram) by group | `sns.histplot()` |
| Smooth distribution by group (KDE) | `sns.kdeplot()` |
| Faceted distributions by group | `sns.displot()` with `col=` or `row=` |

In [118…

```python
%matplotlib inline
plt.rcParams['figure.figsize']=6,2
vis2= sns.boxplot(data=df,x="IncomeGroup",y="InternetUsers",hue="IncomeGrou
plt.legend()
```

Out[118…

```
<matplotlib.legend.Legend at 0x24c633361d0>
```



100
75
50
25
0
InternetUsers
High income
Low income
Upper middle income
Lower middle income
IncomeGroup


In [ ]: