PROJECT : COUNTRY GDP(Gross Domestic Product ANALYSIS

- 1st family - 10 member (1member is earning) rest of eating - Economy will be low and GDP will be low
- 2nd family - 4member(4member earning) - Economy will be high and GDP will be high

COUNTRY ECONONY OR GDP depends on every individual

In [121…
```
import pandas as pd     # To import dataframes
```

In [123…
```
pd.__version__
```

Out[123… '2.2.2'

In [125…
```
df = pd.read_csv (r'D:\Full Stack Data Scientist and AI\March 20 - Pandas Projec
```

In [127…
```
df
```

Out[127…

	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

pd.read_csv -> if reading csv file. if any other file ex. pd.read_excel etc. press tab for more info

In [130…
```
len(df)  #df is an object that stores 195 rows and 5 columns
```

Out[130…
```
195
```

In [132…
```
df.shape  #dimensions
```

Out[132…
```
(195, 5)
```

In [134…
```
df.columns  #Displays columns
```

Out[134…
```
Index(['CountryName', 'CountryCode', 'BirthRate', 'InternetUsers',
       'IncomeGroup'],
      dtype='object')
```

In [136…
```
len(df.columns)  #Displays number of columns
```

Out[136…
```
5
```

In [138…
```
type(df)
```

Out[138…
```
pandas.core.frame.DataFrame
```

In [140…
```
df.info()  #Displays info of dataframe
```

```
<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
```

- We are checking memory as in future we will create app

In [143…
```
df.head()  #Prints top 5 rows
```

Out[143…

	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 [145…
```
df.head(2)  #Prints top 2 rows
```

Out[145…

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

In [147…
```
df.tail()   #Prints last 5 rows
```

Out[147…

	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

In [149…
```
df.tail(2)   #Prints last 2 rows
```

Out[149…

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup
193	Zambia	ZMB	40.471	15.4	Lower middle income
194	Zimbabwe	ZWE	35.715	18.5	Low income

In [151…
```
df.tail(7)   #Prints last 7 rows
```

Out[151…

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup
188	West Bank and Gaza	PSE	30.394	46.6	Lower middle income
189	Samoa	WSM	26.172	15.3	Lower middle 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

Reversing a DataFrame

In [154…
```
df[::-1]  #Reverse the dataset
```

Out[154...

	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 [156...

```
df[: : 20]   #20 stepcount is followed
```

Out[156...

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup
0	Aruba	ABW	10.244	78.9000	High income
20	Belarus	BLR	12.500	54.1700	Upper middle income
40	Costa Rica	CRI	15.022	45.9600	Upper middle income
60	Gabon	GAB	30.555	9.2000	Upper middle income
80	India	IND	20.291	15.1000	Lower middle income
100	Libya	LBY	21.425	16.5000	Upper middle income
120	Mozambique	MOZ	39.705	5.4000	Low income
140	Poland	POL	9.600	62.8492	High income
160	Suriname	SUR	18.455	37.4000	Upper middle income
180	Uruguay	URY	14.374	57.6900	High income

In [158...

```
df[:5]   #0 to 4 rows will be displayed. (n-1) rule is applicable
```

Out[158…

	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 [160…

```
df[6:]
```

Out[160…

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup
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
...	...	...	...	...	...
190	Yemen, Rep.	YEM	32.947	20.0000	Lower middle income
191	South Africa	ZAF	20.850	46.5000	Upper middle income
192	Congo, Dem. Rep.	COD	42.394	2.2000	Low income
193	Zambia	ZMB	40.471	15.4000	Lower middle income
194	Zimbabwe	ZWE	35.715	18.5000	Low income

189 rows × 5 columns

In [162…

```
df[0:200:10]
```

Out[162...

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup
0	Aruba	ABW	10.244	78.900000	High income
10	Azerbaijan	AZE	18.300	58.700000	Upper middle income
20	Belarus	BLR	12.500	54.170000	Upper middle income
30	Canada	CAN	10.900	85.800000	High income
40	Costa Rica	CRI	15.022	45.960000	Upper middle income
50	Ecuador	ECU	21.070	40.353684	Upper middle income
60	Gabon	GAB	30.555	9.200000	Upper middle income
70	Greenland	GRL	14.500	65.800000	High income
80	India	IND	20.291	15.100000	Lower middle income
90	Kazakhstan	KAZ	22.730	54.000000	Upper middle income
100	Libya	LBY	21.425	16.500000	Upper middle income
110	Moldova	MDA	12.141	45.000000	Lower middle income
120	Mozambique	MOZ	39.705	5.400000	Low income
130	Netherlands	NLD	10.200	93.956400	High income
140	Poland	POL	9.600	62.849200	High income
150	Sudan	SDN	33.477	22.700000	Lower middle income
160	Suriname	SUR	18.455	37.400000	Upper middle income
170	Tajikistan	TJK	30.792	16.000000	Lower middle income
180	Uruguay	URY	14.374	57.690000	High income
190	Yemen, Rep.	YEM	32.947	20.000000	Lower middle income

In [164...

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

Out[164...

	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

To get statistics on the columns

In [167...

```
df.describe()  #Descriptive stats
```

Out[167...

	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 [169...
```
df.describe().transpose()  #transpose() converts rows into columns and vice-vers
```

Out[169...

	count	mean	std	min	25%	50%	75%	max
BirthRate	195.0	21.469928	10.605467	7.9	12.1205	19.68	29.7595	49.6610
InternetUsers	195.0	42.076471	29.030788	0.9	14.5200	41.00	66.2250	96.5468

In [171...
```
df.describe().T  #T and transpose() are same, converts rows into columns and vic
```

Out[171...

	count	mean	std	min	25%	50%	75%	max
BirthRate	195.0	21.469928	10.605467	7.9	12.1205	19.68	29.7595	49.6610
InternetUsers	195.0	42.076471	29.030788	0.9	14.5200	41.00	66.2250	96.5468

In [173...
```
df.columns
```

Out[173...
```
Index(['CountryName', 'CountryCode', 'BirthRate', 'InternetUsers',
       'IncomeGroup'],
      dtype='object')
```

Renaming columns of a dataframe

In [176...
```
df.columns = ['a', 'b', 'c', 'd', 'e'] #Changing the column names, list is mutab
```

In [178...
```
df.head(1)
```

Out[178...

	a	b	c	d	e
0	Aruba	ABW	10.244	78.9	High income

Thus, column names are changed. Attributes are renamed.

In [181...
```
df.columns
```

Out[181...
```
Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
```

In [183…
```
df.columns = ['CountryName', 'CountryCode', 'BirthRate', 'InternetUsers',
       'IncomeGroup']   #Putting old columns
```

In [185…
```
df.head(1)
```

Out[185…

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup
0	Aruba	ABW	10.244	78.9	High income

In [187…
```
df[:]
```

Out[187…

	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 [189…
```
df[21:26]
```

Out[189…

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup
21	Belize	BLZ	23.092	33.60	Upper middle income
22	Bermuda	BMU	10.400	95.30	High income
23	Bolivia	BOL	24.236	36.94	Lower middle income
24	Brazil	BRA	14.931	51.04	Upper middle income
25	Barbados	BRB	12.188	73.00	High income

In [191…
```
df[0:5]
```

Out[191…

	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 [193…

```
df[:: 10]  #10 is the step count
```

Out[193…

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup
0	Aruba	ABW	10.244	78.900000	High income
10	Azerbaijan	AZE	18.300	58.700000	Upper middle income
20	Belarus	BLR	12.500	54.170000	Upper middle income
30	Canada	CAN	10.900	85.800000	High income
40	Costa Rica	CRI	15.022	45.960000	Upper middle income
50	Ecuador	ECU	21.070	40.353684	Upper middle income
60	Gabon	GAB	30.555	9.200000	Upper middle income
70	Greenland	GRL	14.500	65.800000	High income
80	India	IND	20.291	15.100000	Lower middle income
90	Kazakhstan	KAZ	22.730	54.000000	Upper middle income
100	Libya	LBY	21.425	16.500000	Upper middle income
110	Moldova	MDA	12.141	45.000000	Lower middle income
120	Mozambique	MOZ	39.705	5.400000	Low income
130	Netherlands	NLD	10.200	93.956400	High income
140	Poland	POL	9.600	62.849200	High income
150	Sudan	SDN	33.477	22.700000	Lower middle income
160	Suriname	SUR	18.455	37.400000	Upper middle income
170	Tajikistan	TJK	30.792	16.000000	Lower middle income
180	Uruguay	URY	14.374	57.690000	High income
190	Yemen, Rep.	YEM	32.947	20.000000	Lower middle income

In [195…

```
df.columns
```

Out[195…

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

In [197…

```
df
```

Out[197…

	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 [199…

```
df[['CountryName', 'CountryCode']]
```

Out[199…

	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 [201…

```
df.isnull()
```

Out[201…

	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 [203…
```
df.isnull().sum()
```

Out[203…
```
CountryName      0
CountryCode      0
BirthRate        0
InternetUsers    0
IncomeGroup      0
dtype: int64
```

Splitting the data into Categorical and Numerical Dataset

In [206…
```
df.dtypes   #This gives the datatype of the columns. df.dtypes is a function
```

Out[206…
```
CountryName       object
CountryCode       object
BirthRate        float64
InternetUsers    float64
IncomeGroup       object
dtype: object
```

In [208…
```
df.columns
```

Out[208…
```
Index(['CountryName', 'CountryCode', 'BirthRate', 'InternetUsers',
       'IncomeGroup'],
      dtype='object')
```

In [210…
```
df_categorical = df[['CountryName', 'CountryCode', 'IncomeGroup']]
df_categorical
```

Out[210...

	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 [212...
```
df_categorical.describe()
```

Out[212...

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

In [214...
```
df_categorical.head()
```

Out[214...

	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 [216...
```
df.describe()
```

Out[216...

	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 [218...

```
df_num = df[['BirthRate', 'InternetUsers']]
df_num.head()
```

Out[218...

	BirthRate	InternetUsers
0	10.244	78.9
1	35.253	5.9
2	45.985	19.1
3	12.877	57.2
4	11.044	88.0

In [220...

```
df_num.describe().transpose()  #Rows to columns and vice versa
```

Out[220...

	count	mean	std	min	25%	50%	75%	max
BirthRate	195.0	21.469928	10.605467	7.9	12.1205	19.68	29.7595	49.6610
InternetUsers	195.0	42.076471	29.030788	0.9	14.5200	41.00	66.2250	96.5468

In [222...

```
df.head()
```

Out[222...

	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 [224...

```
df['IncomeGroup']
```

Out[224…
```
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
Name: IncomeGroup, Length: 195, dtype: object
```

In [226…
```
df.columns
```

Out[226…
```
Index(['CountryName', 'CountryCode', 'BirthRate', 'InternetUsers',
       'IncomeGroup'],
      dtype='object')
```

In [228…
```
['CountryName', 'BirthRate']
```

Out[228…
```
['CountryName', 'BirthRate']
```

In [230…
```
df[['CountryName', 'BirthRate']]

#df is a dataframe that stores entire rows and columns.
#This code gives 2 attribute's info
```

Out[230…

	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 [232…
```
df[['CountryName', 'BirthRate', 'IncomeGroup']]
```

Out[232...

	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 [234...
```
df.head()
```

Out[234...

	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 [236...
```
df['BirthRate']
```

Out[236...
```
0      10.244
1      35.253
2      45.985
3      12.877
4      11.044
        ...
190    32.947
191    20.850
192    42.394
193    40.471
194    35.715
Name: BirthRate, Length: 195, dtype: float64
```

In [238...
```
df[['BirthRate']]
```

Out[238…

	BirthRate
0	10.244
1	35.253
2	45.985
3	12.877
4	11.044
…	…
190	32.947
191	20.850
192	42.394
193	40.471
194	35.715

195 rows × 1 columns

In [240…
```
df.columns
```

Out[240…
```
Index(['CountryName', 'CountryCode', 'BirthRate', 'InternetUsers',
       'IncomeGroup'],
      dtype='object')
```

In [242…
```
df
```

Out[242...

	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 [244...
```
df[4:8]
```

Out[244...

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup
4	United Arab Emirates	ARE	11.044	88.0	High income
5	Argentina	ARG	17.716	59.9	High income
6	Armenia	ARM	13.308	41.9	Lower middle income
7	Antigua and Barbuda	ATG	16.447	63.4	High income

In [246...
```
df[4:8][['CountryName', 'BirthRate']]
```

Out[246...

	CountryName	BirthRate
4	United Arab Emirates	11.044
5	Argentina	17.716
6	Armenia	13.308
7	Antigua and Barbuda	16.447

In [248...
```
df[['CountryName', 'BirthRate']][4:8] #same output as x.y = y.x i.e lhs = rhs
```

Out[248...

	CountryName	BirthRate
4	United Arab Emirates	11.044
5	Argentina	17.716
6	Armenia	13.308
7	Antigua and Barbuda	16.447

In [250...

```
df[['CountryCode','BirthRate','InternetUsers']][4:8] #subet dataframe
```

Out[250...

	CountryCode	BirthRate	InternetUsers
4	ARE	11.044	88.0
5	ARG	17.716	59.9
6	ARM	13.308	41.9
7	ATG	16.447	63.4

In [252...

```
df[['CountryName','BirthRate']].head()
```

Out[252...

	CountryName	BirthRate
0	Aruba	10.244
1	Afghanistan	35.253
2	Angola	45.985
3	Albania	12.877
4	United Arab Emirates	11.044

In [254...

```
df_copy = df [['CountryName','BirthRate']]
df_copy
```

Out[254...

	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 [256...
```
df_copy = df[4:8]
df_copy
```

Out[256...

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup
4	United Arab Emirates	ARE	11.044	88.0	High income
5	Argentina	ARG	17.716	59.9	High income
6	Armenia	ARM	13.308	41.9	Lower middle income
7	Antigua and Barbuda	ATG	16.447	63.4	High income

Mathmetical Operation

- It is performed only on Numerical data and NOT on Categorical data

In [259...
```
df.BirthRate * df.InternetUsers
```

Out[259...
```
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
```

Adding an Extra Column

In [262...
```
# Add a column

df['My_Calc'] = df.BirthRate * df.InternetUsers
```

In [264...
```
df.head()
```

Out[264...

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup	My_Calc
0	Aruba	ABW	10.244	78.9	High income	808.2516
1	Afghanistan	AFG	35.253	5.9	Low income	207.9927
2	Angola	AGO	45.985	19.1	Upper middle income	878.3135
3	Albania	ALB	12.877	57.2	Upper middle income	736.5644
4	United Arab Emirates	ARE	11.044	88.0	High income	971.8720

In [266...
```
#Remove a column

df.drop('myCalc',axis = 1)  # Axis 0 = Rows, Axis 1 = Columns
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[266], line 3
      1 #Remove a column
----> 3 df.drop('myCalc',axis = 1)

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:5581, in DataFrame.drop(self, labels, axis, index, columns, level, inplace, errors)
   5433 def drop(
   5434     self,
   5435     labels: IndexLabel | None = None,
   (...)
   5442     errors: IgnoreRaise = "raise",
   5443 ) -> DataFrame | None:
   5444     """
   5445     Drop specified labels from rows or columns.
   5446
   (...)
   5579             weight  1.0     0.8
   5580     """
-> 5581     return super().drop(
   5582         labels=labels,
   5583         axis=axis,
   5584         index=index,
   5585         columns=columns,
   5586         level=level,
   5587         inplace=inplace,
   5588         errors=errors,
   5589     )

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4788, in NDFrame.drop(self, labels, axis, index, columns, level, inplace, errors)
   4786 for axis, labels in axes.items():
   4787     if labels is not None:
-> 4788         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4790 if inplace:
   4791     self._update_inplace(obj)

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4830, in NDFrame._drop_axis(self, labels, axis, level, errors, only_slice)
   4828         new_axis = axis.drop(labels, level=level, errors=errors)
   4829     else:
-> 4830         new_axis = axis.drop(labels, errors=errors)
   4831     indexer = axis.get_indexer(new_axis)
   4833 # Case for non-unique axis
   4834 else:

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:7070, in Index.drop(self, labels, errors)
   7068 if mask.any():
   7069     if errors != "ignore":
-> 7070         raise KeyError(f"{labels[mask].tolist()} not found in axis")
   7071     indexer = indexer[~mask]
   7072 return self.delete(indexer)

KeyError: "['myCalc'] not found in axis"
```

In [268…

```
df = df.drop('myCalc',axis = 1) # Axis 0 = Rows, Axis 1 = Columns
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[268], line 1
----> 1 df = df.drop('myCalc',axis = 1)

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:5581, in DataFrame.drop(s
elf, labels, axis, index, columns, level, inplace, errors)
   5433 def drop(
   5434     self,
   5435     labels: IndexLabel | None = None,
   (...)
   5442     errors: IgnoreRaise = "raise",
   5443 ) -> DataFrame | None:
   5444     """
   5445     Drop specified labels from rows or columns.
   5446
   (...)
   5579             weight  1.0     0.8
   5580     """
-> 5581     return super().drop(
   5582         labels=labels,
   5583         axis=axis,
   5584         index=index,
   5585         columns=columns,
   5586         level=level,
   5587         inplace=inplace,
   5588         errors=errors,
   5589     )

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4788, in NDFrame.drop(s
elf, labels, axis, index, columns, level, inplace, errors)
   4786 for axis, labels in axes.items():
   4787     if labels is not None:
-> 4788         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4790 if inplace:
   4791     self._update_inplace(obj)

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4830, in NDFrame._drop_
axis(self, labels, axis, level, errors, only_slice)
   4828         new_axis = axis.drop(labels, level=level, errors=errors)
   4829     else:
-> 4830         new_axis = axis.drop(labels, errors=errors)
   4831     indexer = axis.get_indexer(new_axis)
   4833 # Case for non-unique axis
   4834 else:

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:7070, in Index.dro
p(self, labels, errors)
   7068 if mask.any():
   7069     if errors != "ignore":
-> 7070         raise KeyError(f"{labels[mask].tolist()} not found in axis")
   7071     indexer = indexer[~mask]
   7072 return self.delete(indexer)

KeyError: "['myCalc'] not found in axis"
```

In [270…

```
df.head()
```

Out[270...

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup	My_Calc
0	Aruba	ABW	10.244	78.9	High income	808.2516
1	Afghanistan	AFG	35.253	5.9	Low income	207.9927
2	Angola	AGO	45.985	19.1	Upper middle income	878.3135
3	Albania	ALB	12.877	57.2	Upper middle income	736.5644
4	United Arab Emirates	ARE	11.044	88.0	High income	971.8720

In [272...
```
df.columns  # New coloumn is added -> 'My_Calc'
```

Out[272...
```
Index(['CountryName', 'CountryCode', 'BirthRate', 'InternetUsers',
       'IncomeGroup', 'My_Calc'],
      dtype='object')
```

In [274...
```
df.columns[3:4]  #3 slice 4 means 'InternetUsers' column
```

Out[274...
```
Index(['InternetUsers'], dtype='object')
```

In [276...
```
df.columns[2]
```

Out[276...
```
'BirthRate'
```

In [278...
```
df['InternetUsers']
```

Out[278...
```
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 [280...
```
df[3:7]
```

Out[280...

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup	My_Calc
3	Albania	ALB	12.877	57.2	Upper middle income	736.5644
4	United Arab Emirates	ARE	11.044	88.0	High income	971.8720
5	Argentina	ARG	17.716	59.9	High income	1061.1884
6	Armenia	ARM	13.308	41.9	Lower middle income	557.6052

In [282…
```
df[30:40]
```

Out[282…

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup	My_Calc
30	Canada	CAN	10.900	85.80	High income	935.2200
31	Switzerland	CHE	10.200	86.34	High income	880.6680
32	Chile	CHL	13.385	66.50	High income	890.1025
33	China	CHN	12.100	45.80	Upper middle income	554.1800
34	Cote d'Ivoire	CIV	37.320	8.40	Lower middle income	313.4880
35	Cameroon	CMR	37.236	6.40	Lower middle income	238.3104
36	Congo, Rep.	COG	37.011	6.60	Lower middle income	244.2726
37	Colombia	COL	16.076	51.70	Upper middle income	831.1292
38	Comoros	COM	34.326	6.50	Low income	223.1190
39	Cabo Verde	CPV	21.625	37.50	Lower middle income	810.9375

Condition Checking - Applying Filters

In [285…
```
df.InternetUsers<2 #we are checking given condition if its correct true or false

#Ex. Checking families where there are less than 2 internet users
```

Out[285…
```
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 [287…
```
df
```

Out[287…

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup	My_Calc
0	Aruba	ABW	10.244	78.9	High income	808.2516
1	Afghanistan	AFG	35.253	5.9	Low income	207.9927
2	Angola	AGO	45.985	19.1	Upper middle income	878.3135
3	Albania	ALB	12.877	57.2	Upper middle income	736.5644
4	United Arab Emirates	ARE	11.044	88.0	High income	971.8720
...	...	...	...	...	...	...
190	Yemen, Rep.	YEM	32.947	20.0	Lower middle income	658.9400
191	South Africa	ZAF	20.850	46.5	Upper middle income	969.5250
192	Congo, Dem. Rep.	COD	42.394	2.2	Low income	93.2668
193	Zambia	ZMB	40.471	15.4	Lower middle income	623.2534
194	Zimbabwe	ZWE	35.715	18.5	Low income	660.7275

195 rows × 6 columns

In [289…
```
Filter = df.InternetUsers < 2   #Filter variable is created
Filter
```

Out[289…
```
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 [291…
```
df[Filter] # It will pull out countries where lowest sale has happened of a prod
```

Out[291…

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup	My_Calc
11	Burundi	BDI	44.151	1.3	Low income	57.3963
52	Eritrea	ERI	34.800	0.9	Low income	31.3200
55	Ethiopia	ETH	32.925	1.9	Low income	62.5575
64	Guinea	GIN	37.337	1.6	Low income	59.7392
117	Myanmar	MMR	18.119	1.6	Lower middle income	28.9904
127	Niger	NER	49.661	1.7	Low income	84.4237
154	Sierra Leone	SLE	36.729	1.7	Low income	62.4393
156	Somalia	SOM	43.891	1.5	Low income	65.8365
172	Timor-Leste	TLS	35.755	1.1	Lower middle income	39.3305

In [293…
```
len(df[Filter]) # It will display the number of countries with low income
```

Out[293… 9

- BPL - Below Poverty Line (If a family income is less than Rs. 50,000)
- APL - Above Poverty line (If a family income is more than Rs. 1,00,000)

Thus, government has to support BPL families

In [296…
```
df
```

Out[296...

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup	My_Calc
0	Aruba	ABW	10.244	78.9	High income	808.2516
1	Afghanistan	AFG	35.253	5.9	Low income	207.9927
2	Angola	AGO	45.985	19.1	Upper middle income	878.3135
3	Albania	ALB	12.877	57.2	Upper middle income	736.5644
4	United Arab Emirates	ARE	11.044	88.0	High income	971.8720
...	...	...	...	...	...	...
190	Yemen, Rep.	YEM	32.947	20.0	Lower middle income	658.9400
191	South Africa	ZAF	20.850	46.5	Upper middle income	969.5250
192	Congo, Dem. Rep.	COD	42.394	2.2	Low income	93.2668
193	Zambia	ZMB	40.471	15.4	Lower middle income	623.2534
194	Zimbabwe	ZWE	35.715	18.5	Low income	660.7275

195 rows × 6 columns

Out of 9 countries, which country government want to support, let's put another condition

In [299...
```
df[Filter].BirthRate>40  #This tells out of 9 countries, where birthrate > 40
```

Out[299...
```
11      True
52     False
55     False
64     False
117    False
127     True
154    False
156     True
172    False
Name: BirthRate, dtype: bool
```

In [301...
```
df.BirthRate>40
```

Out[301…
```
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 [303…
```
Filter2 = df.BirthRate>40
```

In [305…
```
Filter2
```

Out[305…
```
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 [307…
```
df[Filter2]
```

Out[307…

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup	My_Calc
2	Angola	AGO	45.985	19.1	Upper middle income	878.3135
11	Burundi	BDI	44.151	1.3	Low income	57.3963
14	Burkina Faso	BFA	40.551	9.1	Low income	369.0141
65	Gambia, The	GMB	42.525	14.0	Low income	595.3500
115	Mali	MLI	44.138	3.5	Low income	154.4830
127	Niger	NER	49.661	1.7	Low income	84.4237
128	Nigeria	NGA	40.045	38.0	Lower middle income	1521.7100
156	Somalia	SOM	43.891	1.5	Low income	65.8365
167	Chad	TCD	45.745	2.3	Low income	105.2135
178	Uganda	UGA	43.474	16.2	Low income	704.2788
192	Congo, Dem. Rep.	COD	42.394	2.2	Low income	93.2668
193	Zambia	ZMB	40.471	15.4	Lower middle income	623.2534

In [309…
```
len(df[Filter2])  #12 countries have birthrate > 40
```

Out[309…
```
12
```

In [311…
```
#Filter and Filter2
Filter & Filter2  #And operator is used -> True and True is True, rest all false
```

Out[311…
```
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 [313…
```
df[Filter & Filter2]

#Display countries with low income and uneducated people, government has to supp
```

Out[313…

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup	My_Calc
11	Burundi	BDI	44.151	1.3	Low income	57.3963
127	Niger	NER	49.661	1.7	Low income	84.4237
156	Somalia	SOM	43.891	1.5	Low income	65.8365

Thus, this is how Pandas Framework helps in finding the business insights and the true outcome of the Project

In [316…
```
df[(df.BirthRate > 40) & (df.InternetUsers < 2)]   #Same as df[Filter & Filter2]
```

Out[316…

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup	My_Calc
11	Burundi	BDI	44.151	1.3	Low income	57.3963
127	Niger	NER	49.661	1.7	Low income	84.4237
156	Somalia	SOM	43.891	1.5	Low income	65.8365

In [318…
```
df.head()
```

Out[318…

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup	My_Calc
0	Aruba	ABW	10.244	78.9	High income	808.2516
1	Afghanistan	AFG	35.253	5.9	Low income	207.9927
2	Angola	AGO	45.985	19.1	Upper middle income	878.3135
3	Albania	ALB	12.877	57.2	Upper middle income	736.5644
4	United Arab Emirates	ARE	11.044	88.0	High income	971.8720

In [320…

```
df[df.IncomeGroup == 'Low income']
```

Out[320...

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup	My_Calc
1	Afghanistan	AFG	35.253	5.90	Low income	207.99270
11	Burundi	BDI	44.151	1.30	Low income	57.39630
13	Benin	BEN	36.440	4.90	Low income	178.55600
14	Burkina Faso	BFA	40.551	9.10	Low income	369.01410
29	Central African Republic	CAF	34.076	3.50	Low income	119.26600
38	Comoros	COM	34.326	6.50	Low income	223.11900
52	Eritrea	ERI	34.800	0.90	Low income	31.32000
55	Ethiopia	ETH	32.925	1.90	Low income	62.55750
64	Guinea	GIN	37.337	1.60	Low income	59.73920
65	Gambia, The	GMB	42.525	14.00	Low income	595.35000
66	Guinea-Bissau	GNB	37.503	3.10	Low income	116.25930
77	Haiti	HTI	25.345	10.60	Low income	268.65700
93	Cambodia	KHM	24.462	6.80	Low income	166.34160
99	Liberia	LBR	35.521	3.20	Low income	113.66720
111	Madagascar	MDG	34.686	3.00	Low income	104.05800
115	Mali	MLI	44.138	3.50	Low income	154.48300
120	Mozambique	MOZ	39.705	5.40	Low income	214.40700
123	Malawi	MWI	39.459	5.05	Low income	199.26795
127	Niger	NER	49.661	1.70	Low income	84.42370
132	Nepal	NPL	20.923	13.30	Low income	278.27590
148	Rwanda	RWA	32.689	9.00	Low income	294.20100
154	Sierra Leone	SLE	36.729	1.70	Low income	62.43930
156	Somalia	SOM	43.891	1.50	Low income	65.83650
158	South Sudan	SSD	37.126	14.10	Low income	523.47660
167	Chad	TCD	45.745	2.30	Low income	105.21350
168	Togo	TGO	36.080	4.50	Low income	162.36000
177	Tanzania	TZA	39.518	4.40	Low income	173.87920
178	Uganda	UGA	43.474	16.20	Low income	704.27880
192	Congo, Dem. Rep.	COD	42.394	2.20	Low income	93.26680
194	Zimbabwe	ZWE	35.715	18.50	Low income	660.72750

== for Filtering Records

In [323…
```
df[df.IncomeGroup == 'High income']
```

Out[323…

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup	My_Calc
0	Aruba	ABW	10.244	78.90	High income	808.25160
4	United Arab Emirates	ARE	11.044	88.00	High income	971.87200
5	Argentina	ARG	17.716	59.90	High income	1061.18840
7	Antigua and Barbuda	ATG	16.447	63.40	High income	1042.73980
8	Australia	AUS	13.200	83.00	High income	1095.60000
...	...	...	...	...	...	...
174	Trinidad and Tobago	TTO	14.590	63.80	High income	930.84200
180	Uruguay	URY	14.374	57.69	High income	829.23606
181	United States	USA	12.500	84.20	High income	1052.50000
184	Venezuela, RB	VEN	19.842	54.90	High income	1089.32580
185	Virgin Islands (U.S.)	VIR	10.700	45.30	High income	484.71000

67 rows × 6 columns

How to get the unique categories

- unique()
- nunique()

In [326…
```
df.IncomeGroup.unique()    #Prints Unique Categories in a column
```

Out[326…
```
array(['High income', 'Low income', 'Upper middle income',
       'Lower middle income'], dtype=object)
```

In [328…
```
df.IncomeGroup.nunique()   #Prints Number of Unique Category
```

Out[328… 4

Uptill here we analysed the Python Dataframe or Dataset

Introduction to SeaBorn

- Seaborn is very powerfull visualization (statistical data visualization) package in python
- Matplotlib - Python Library for Visualization
- Seaborn - Python Library for Advanced Visualization | Statistic Visualization

In [333…
```
import matplotlib.pyplot as plt # visulization
import seaborn as sns # distribution visualization

%matplotlib inline
plt.rcParams['figure.figsize'] = 8,4  # 8 = width and 4 = height of the graph

#import warnings
#warnings.filterwarnings('ignore')  #OS error
```

In [335…
```
sns.__version__
```

Out[335…
```
'0.13.2'
```

In [337…
```
df.head()
```

Out[337…

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup	My_Calc
0	Aruba	ABW	10.244	78.9	High income	808.2516
1	Afghanistan	AFG	35.253	5.9	Low income	207.9927
2	Angola	AGO	45.985	19.1	Upper middle income	878.3135
3	Albania	ALB	12.877	57.2	Upper middle income	736.5644
4	United Arab Emirates	ARE	11.044	88.0	High income	971.8720

In [339…
```
df['InternetUsers']
```

Out[339…
```
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 [341…
```
df[['InternetUsers']]
```

Out[341...

	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

Try to read the libraries -

- https://pypi.org/project/seaborn/
- https://github.com/mwaskom/seaborn
- https://github.com/mwaskom/seaborn/blob/master/seaborn/_statistics.py

- Ex. When booking flight or train for next month, Machine learning predicts the probability of ticket confirmation percentage. Based on this, we pay the fee.
- But sometimes, we pay the fee still ticket doesn't get confirmed.
- It is because Machine learning model does not predicts correctly.
- 7 years back, there was nothing like probability, because there was No Machine Learning.
- JAVA, C, C++, C# was there but they didn't fix it because, data is growing, technology also growing as well as the thought process.
- How does the machine predicts ? -> Through **Bayesian Theorem**-> This comes under **Naive Bayes Algorithm**
- When this concept was used in IRCTIC and Flights, their business increased by multi billion rupees. The biggest revenue in government system is this platform - transportation.

In [345...

```
# Distributions:
vis1 = plt.distplot(df["InternetUsers"])
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[345], line 2
      1 # Distributions:
----> 2 vis1 = plt.distplot(df["InternetUsers"])

AttributeError: module 'matplotlib.pyplot' has no attribute 'distplot'
```

In [347…

```
# Distributions:
vis1 = sns.distplot(df["InternetUsers"])
```

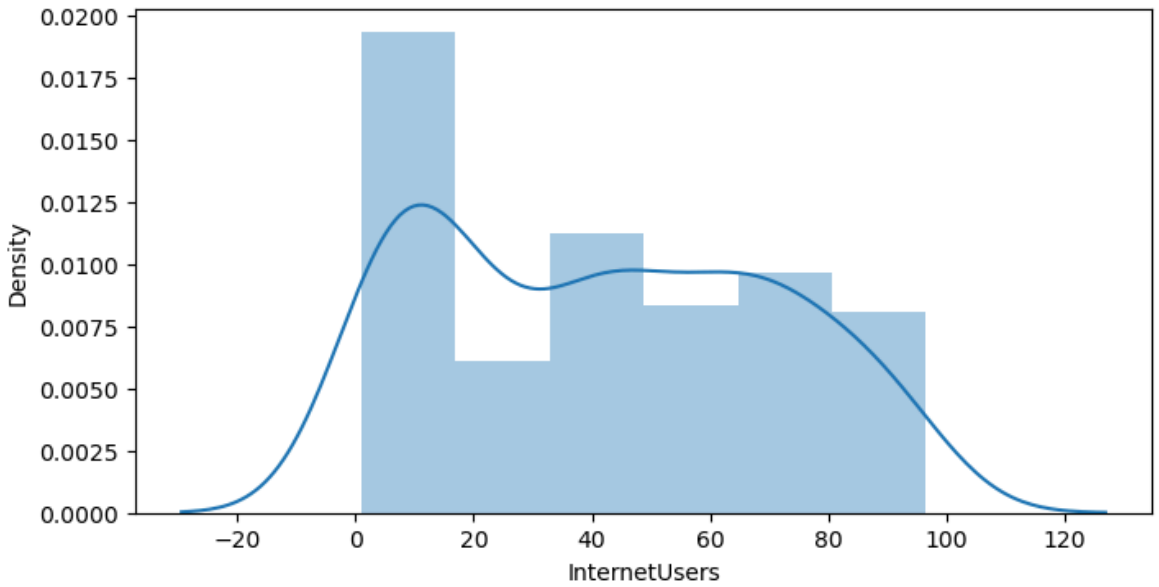
```
C:\Users\kirti\AppData\Local\Temp\ipykernel_16080\1307174048.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  vis1 = sns.distplot(df["InternetUsers"])
```

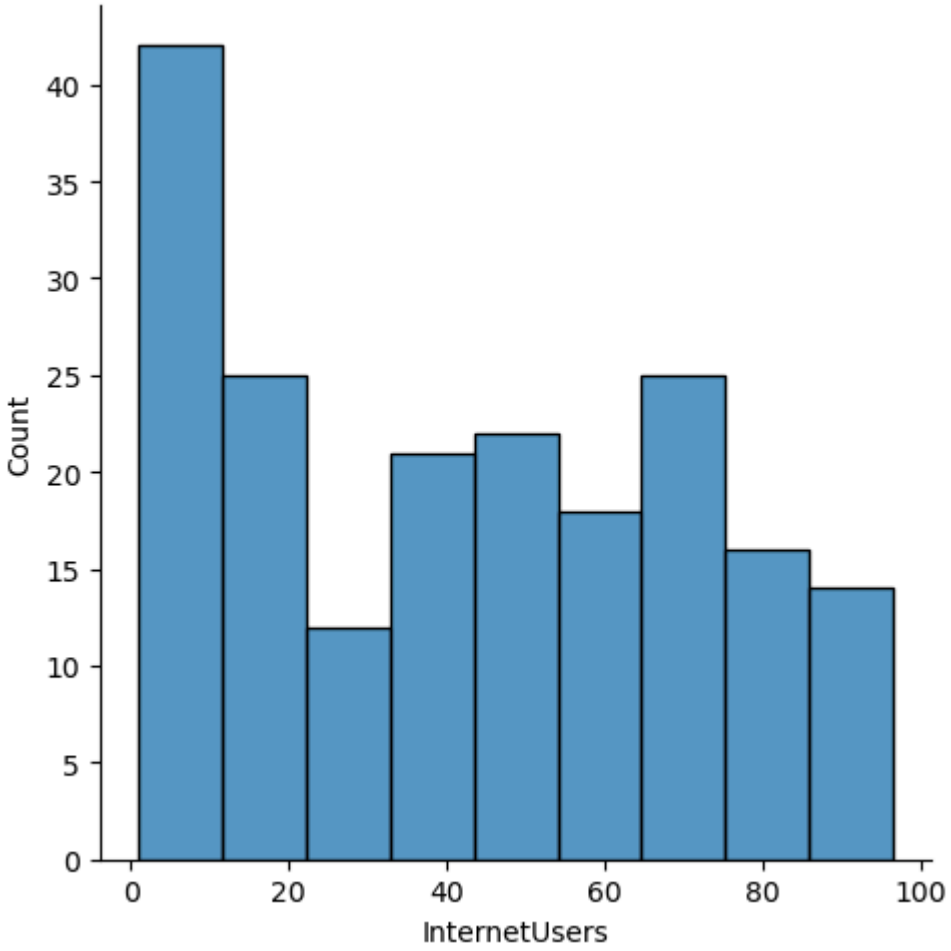
Difference between distplot and displot is ->

- Distplot is Advance Plot - Distribution comes from statistics

In [350…

```
# Distributions:
vis1 = sns.displot(df["InternetUsers"])  #Univariate Analysis
plt.show(vis1)
```





Minimum point of Internet users is 0.9 and maximum is 96 in the Excel sheet

- Hightest Internet user distribution in graph 1 is between 10 - 15
- Lowest Internet user distribution in graph 1 is 96
- Plotting the Graph using 1 variable is called Univariate Analysis

Uni-variate Analysis -> Plotting the graph using 1 variable is called Univariate Analysis in Statistical Term.

How to detect Univariate graph?

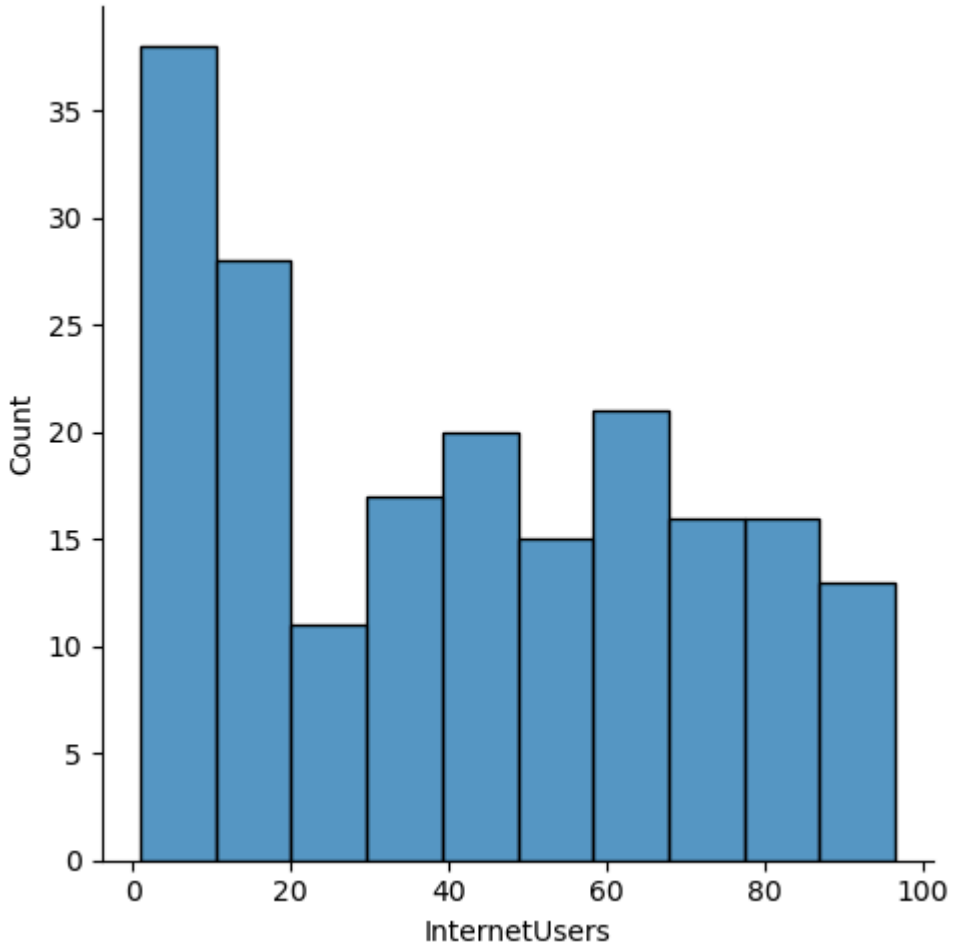
Bi-variate Analysis -> Plotting the graph using 2 variables is called Bivariate Analysis in Statistical Term.

How to detect bi-variate graph? Based on Boxplot

Multi-variate Analysis -> Plotting the graph using multiple variable is called Multivariate Analysis in Statistical Term.

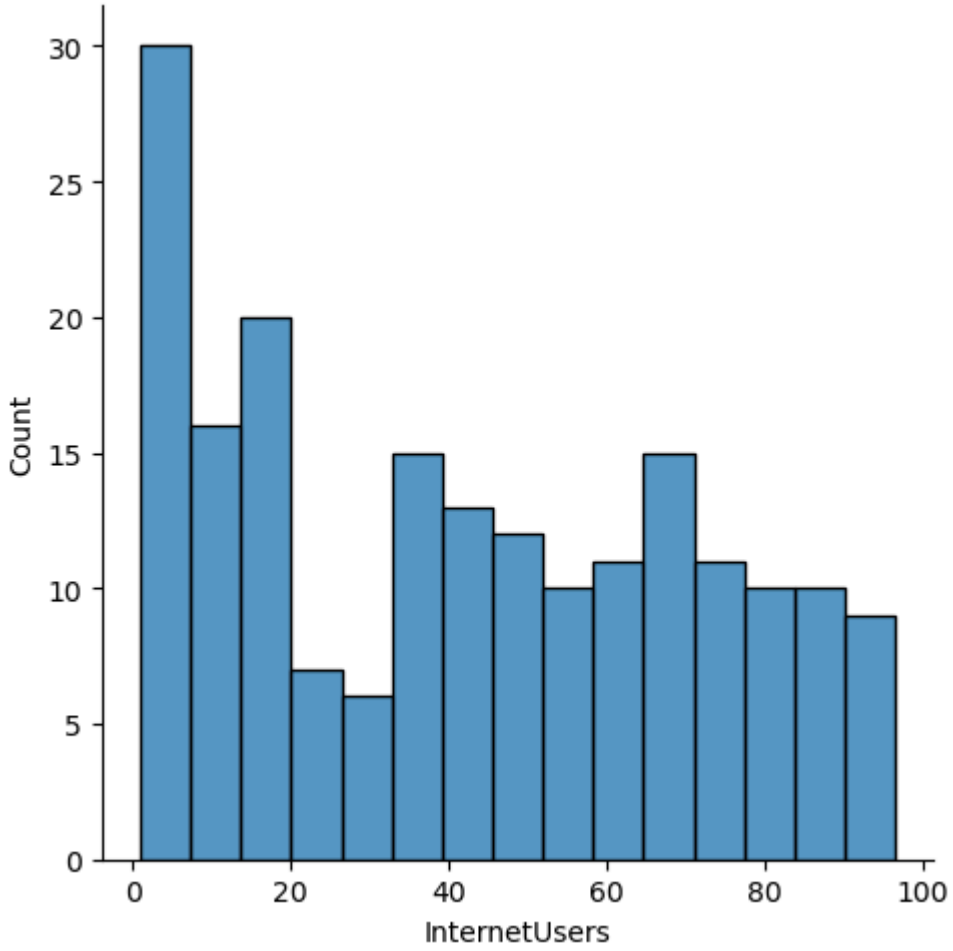
In [355…

```
vis1 = sns.displot(df["InternetUsers"], bins=10)
plt.show(vis1)
```



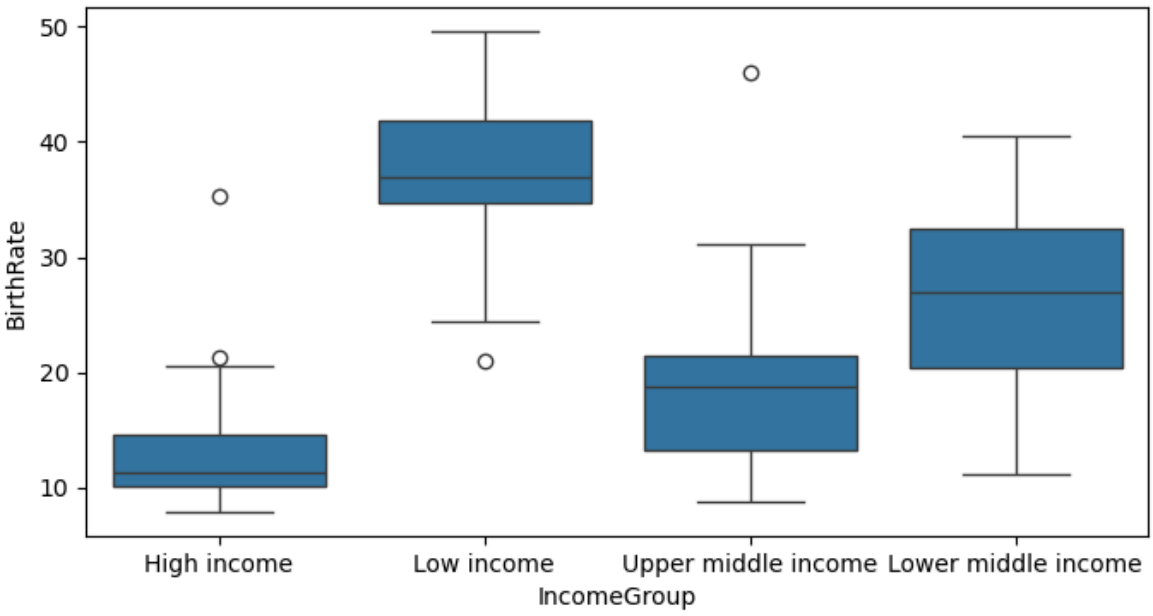
In [357…

```
vis1 = sns.displot(df["InternetUsers"], bins=15)  #15 means 15 bars
plt.show(vis1)
```



In [359…

```
#BOX PLOTS:
vis2 = sns.boxplot(data = df, x="IncomeGroup", y='BirthRate')  #Bi-Variate Analy
plt.show(vis2)

#boxplot -> Name of the graph
```



Meaning of above graph -

x="IncomeGroup" We have 4 income groups -

1. High Income
2. Low Income
3. Upper Middle Income
4. Lower Middle Income

y='BirthRate'

1. Minimum starting point is 7.9
2. Maximum starting point is 35.362

- This bottom line indicates lowest number.
- The top line indicates the maximum number but it is marked below the maximum number on the graph.
- The between box is mean/median
- But in the excel data set 35 is not neighbour to 21.
- In statistics we have concept called **Outlier**
- Technically Outlier is also called **Anomaly Detection**
- In statistics, outlier is a data point which is very far from other observations.

In the above graph, there are 3 outliers. -Ex. 10, 20, 30, 5000 (5000 is the outlier)

- In above Graph

1.) Which Income group has highest Birth rate?

- Ans. Low Income

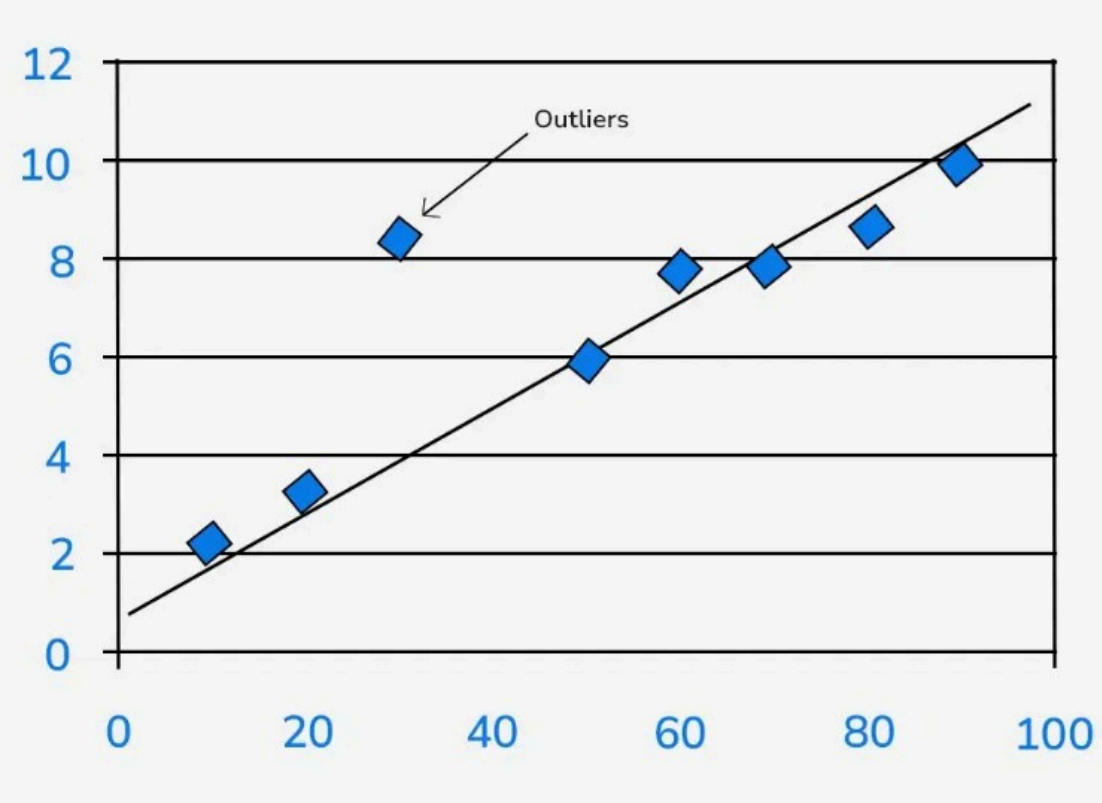
2.) Which Income group has lowest Birth rate?

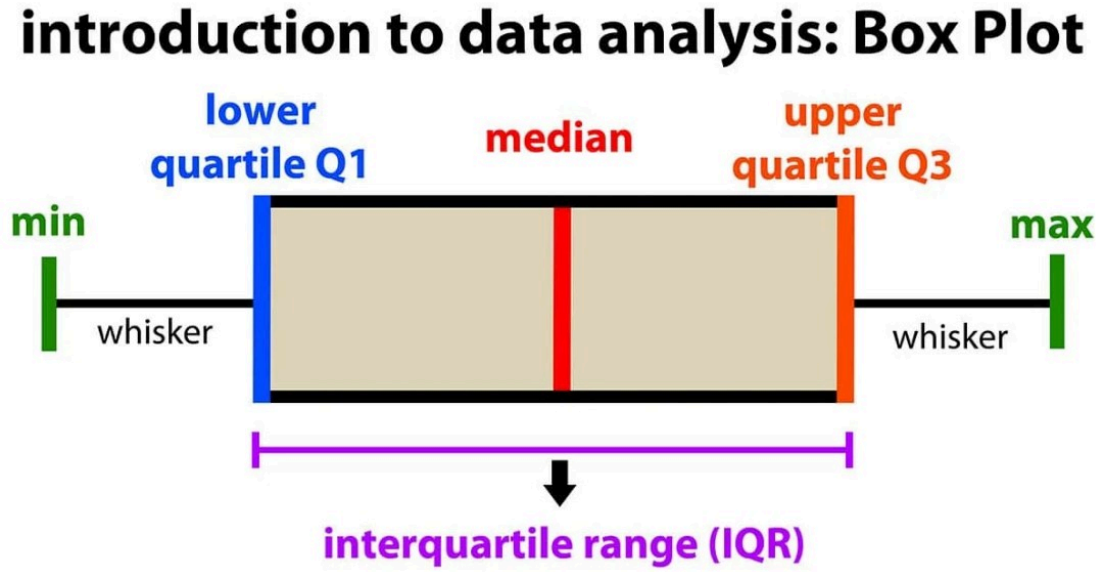
- Ans. High Income

3.) Which Income group has highest average Birth rate?

- Ans. Low Income

4.) Which Income group has lowest average Birth rate?

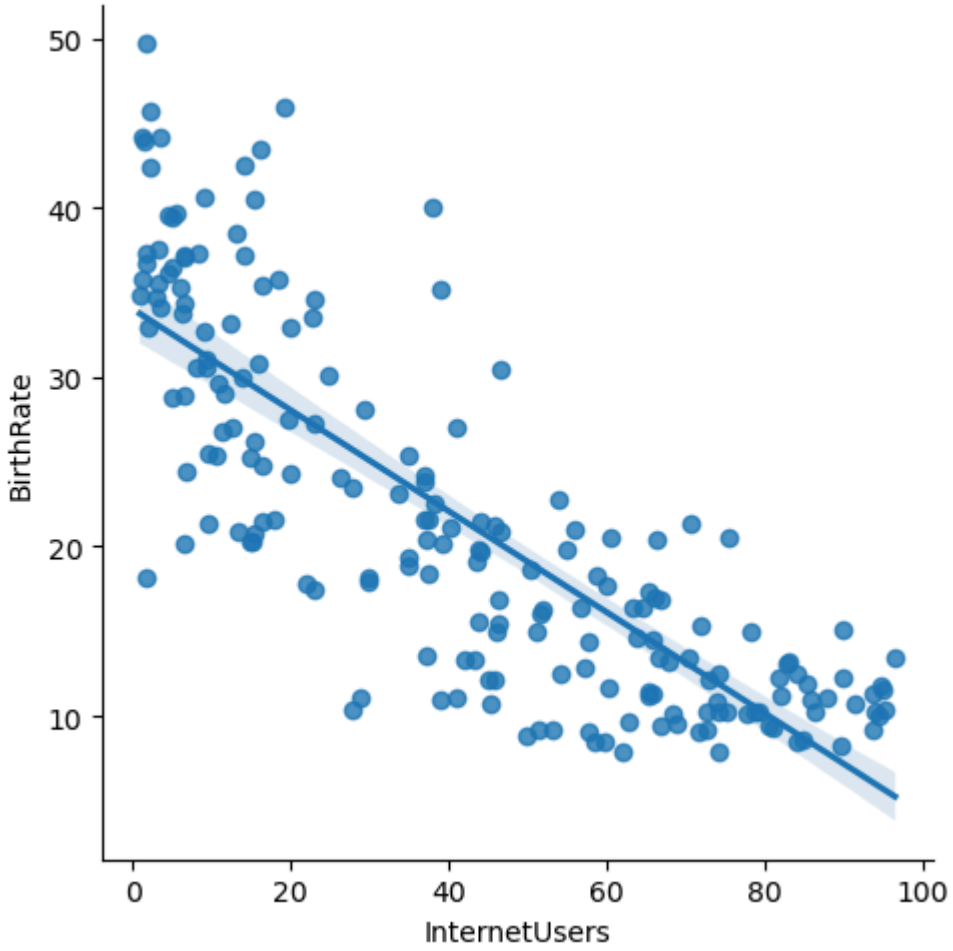




Visualizing with Seaborn

In [366…

```
#lm - linear model

vis3 = sns.lmplot(data = df, x = 'InternetUsers', y = 'BirthRate') # Bi-Variate
plt.show(vis3)
```

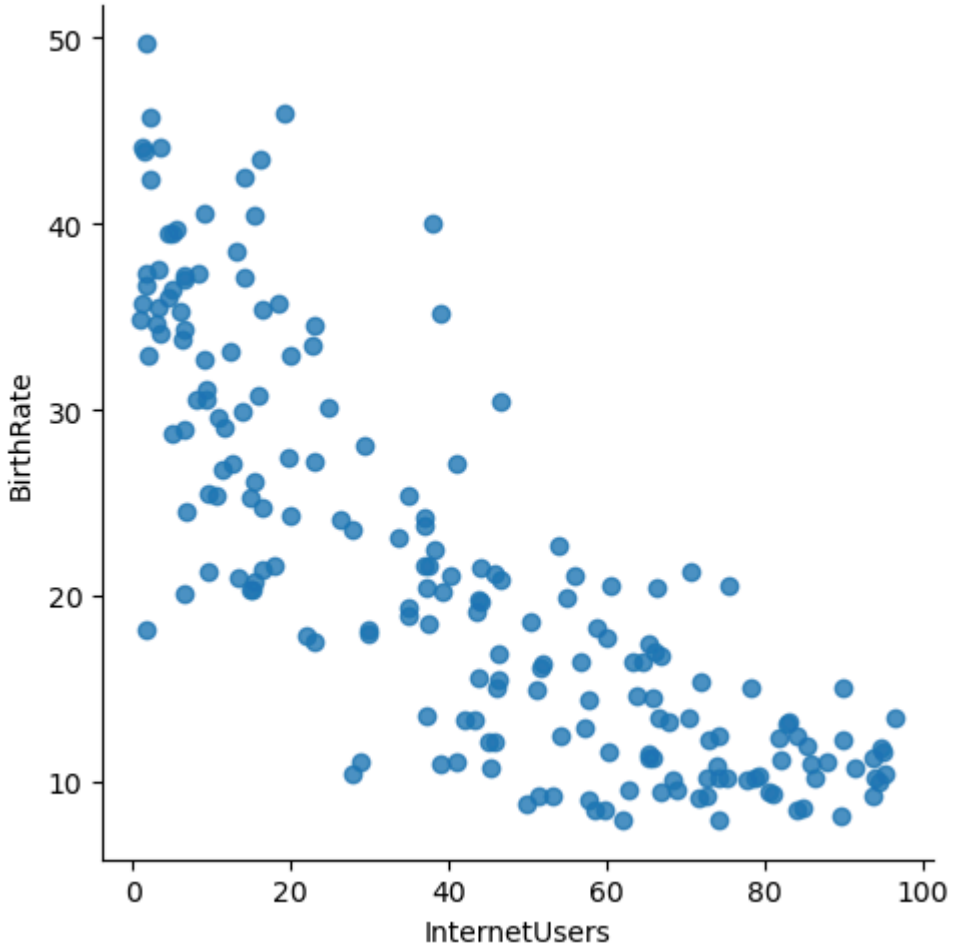


every record in excel is called data point

In above graph, above data points are created from the excel, based on columns

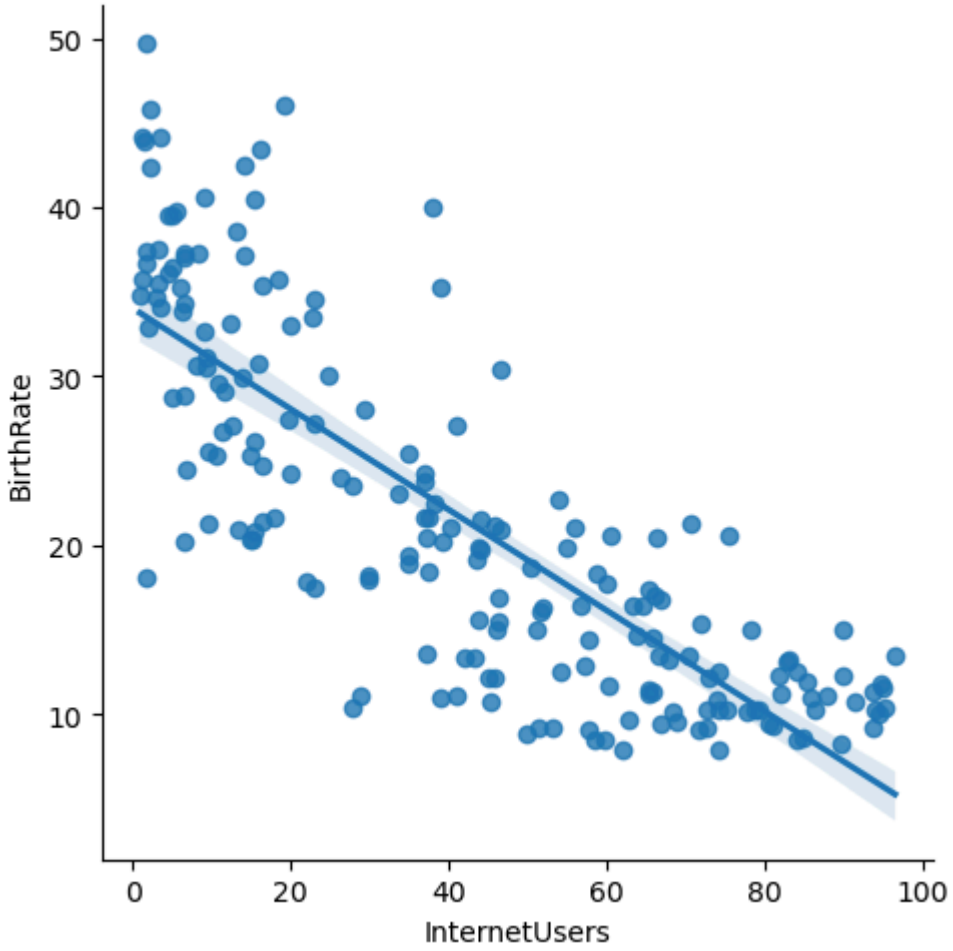
In [369…

```
#fit_reg = False, means no line

vis3 = sns.lmplot(data = df,x = 'InternetUsers', y = 'BirthRate', fit_reg = Fals
plt.show(vis3)
```



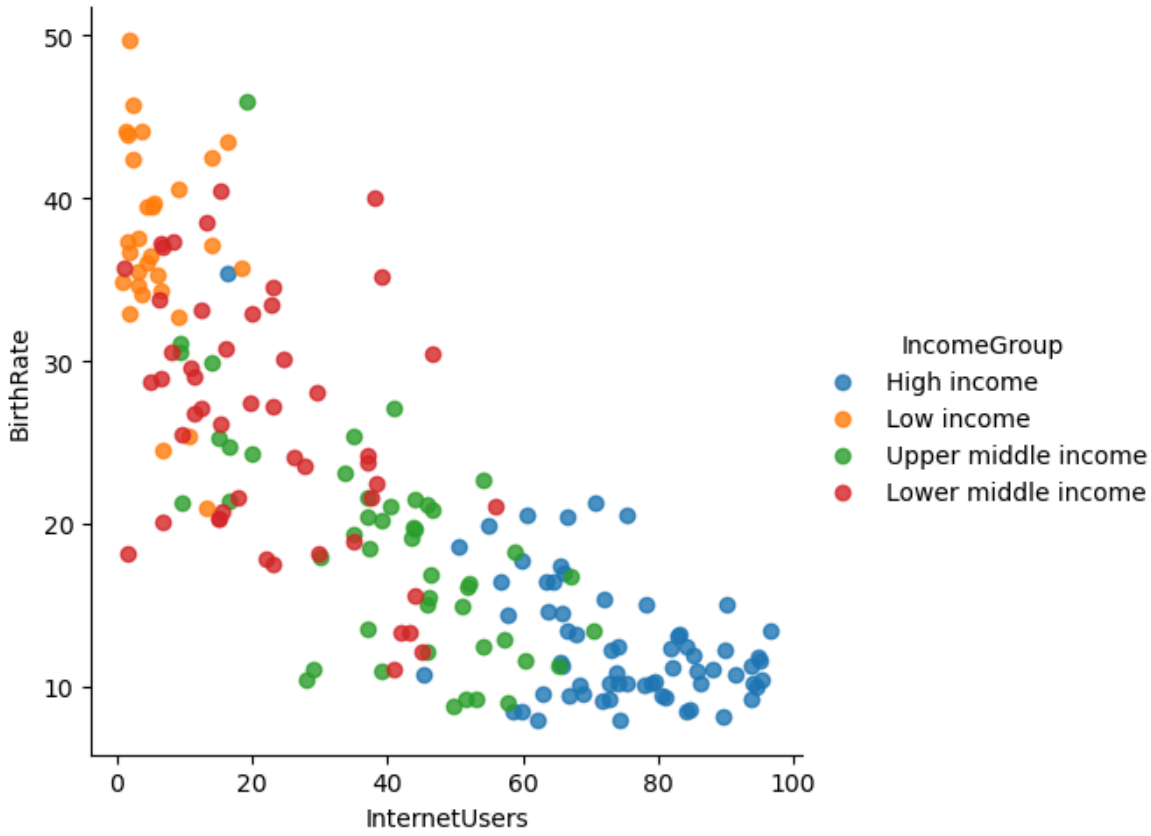
In [371…

```
#fit_reg = True, means there will be line

vis3 = sns.lmplot(data = df,x = 'InternetUsers', y = 'BirthRate', fit_reg = True
plt.show(vis3)
```



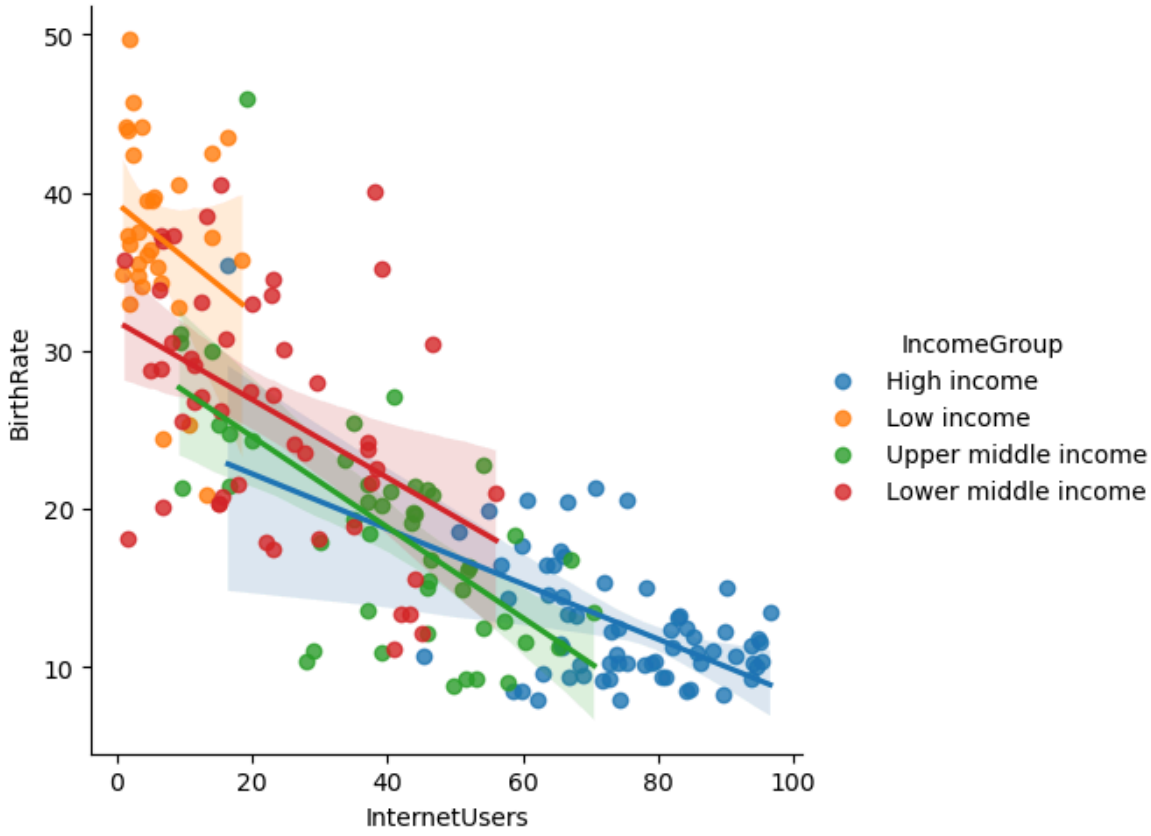
In [373…

```
vis4 = sns.lmplot(data = df,x = 'InternetUsers', y = 'BirthRate',
                  fit_reg = False,hue = 'IncomeGroup') #hue - parameter for colo
plt.show(vis4)

#hue is just like Legend in Numpy, which displays color and label.
```



In [375…

```
vis4 = sns.lmplot(data = df,x = 'InternetUsers', y = 'BirthRate',
                  fit_reg = True ,hue = 'IncomeGroup') #hue - parameter for colo
plt.show(vis4)

#hue is just like Legend in Numpy, which displays color and label.
```



1. Which internet users have lowest birthrate?

Ans. High Income

2. Which internet users have highest birthrate?

   Ans. Low Income

In this section we learned :

1. Importing data into python
2. Dataframe via panda
3. exploring datasets: head()tail()info()describe()
4. Renaming columns
5. subsetting dataframes
6. Basic operations with dataframe
7. Filtering data frames
8. Seaborn introduction -- .distplot | .boxplot | .lmplot(fit_reg) | outlier | hue parameter
9. Univariate, Bivariate, Multivariate analysis

In [ ]: