In [30]:
```
import pandas as pd
```

In [31]:
```
pd.__version__
```

Out[31]:
```
'2.3.0'
```

In [32]:
```
df=pd.read_csv(r"D:\Data Science with AI\Data Science With AI\9th,10th - july-Pa
```

In [33]:
```
df
```

Out[33]:

	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 [34]:
```
id(df)
```

Out[34]:
```
2031496396384
```

In [35]:
```
len(df)
```

Out[35]:
```
195
```

In [36]:
```
df.columns
```

Out[36]:
```
Index(['CountryName', 'CountryCode', 'BirthRate', 'InternetUsers',
       'IncomeGroup'],
      dtype='object')
```

In [37]:
```
len(df.columns)
```

Out[37]: 5

In [38]:
```
df.isnull()
```

Out[38]:

	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 [39]:
```
df.isna()
```

Out[39]:

	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 [40]:
```
df.isnull()
```

Out[40]:

	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 [41]:
```
df.isnull()
```

Out[41]:

	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 [42]:
```
df.isnull().sum()
```

Out[42]:
```
CountryName      0
CountryCode      0
BirthRate        0
InternetUsers    0
IncomeGroup      0
dtype: int64
```

In [43]:
```
df.isna().sum()
```

Out[43]:
```
CountryName      0
CountryCode      0
BirthRate        0
InternetUsers    0
IncomeGroup      0
dtype: int64
```

In [44]:
```
df.head()
```

Out[44]:

	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 [45]:
```
df.tail()
```

Out[45]:

	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 [46]:
```
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 [47]:
```
df[:]
```

Out[47]:

	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 [48]:
```
df[1:11]
```

Out[48]:

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup
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 [49]:
```
df[::-1]
```

Out[49]:

	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 [50]:
```
df[1:100:10]
```

Out[50]:

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup
1	Afghanistan	AFG	35.253	5.9000	Low income
11	Burundi	BDI	44.151	1.3000	Low income
21	Belize	BLZ	23.092	33.6000	Upper middle income
31	Switzerland	CHE	10.200	86.3400	High income
41	Cuba	CUB	10.400	27.9300	Upper middle income
51	Egypt, Arab Rep.	EGY	28.032	29.4000	Lower middle income
61	United Kingdom	GBR	12.200	89.8441	High income
71	Guatemala	GTM	27.465	19.7000	Lower middle income
81	Ireland	IRL	15.000	78.2477	High income
91	Kenya	KEN	35.194	39.0000	Lower middle income

In [51]:
```
df[10:21]
```

Out[51]:

	CountryName	CountryCode	BirthRate	InternetUsers	IncomeGroup
10	Azerbaijan	AZE	18.300	58.70000	Upper middle income
11	Burundi	BDI	44.151	1.30000	Low income
12	Belgium	BEL	11.200	82.17020	High income
13	Benin	BEN	36.440	4.90000	Low income
14	Burkina Faso	BFA	40.551	9.10000	Low income
15	Bangladesh	BGD	20.142	6.63000	Lower middle income
16	Bulgaria	BGR	9.200	53.06150	Upper middle income
17	Bahrain	BHR	15.040	90.00004	High income
18	Bahamas, The	BHS	15.339	72.00000	High income
19	Bosnia and Herzegovina	BIH	9.062	57.79000	Upper middle income
20	Belarus	BLR	12.500	54.17000	Upper middle income

In [52]:
```
df.head(2)
```

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

In [53]:
```
df.describe() # by default describe function can write only numerical records
```

Out[53]:

	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 [54]:
```
df.head(1)
```

Out[54]:

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

In [55]:
```
df['CountryName']
```

Out[55]:
```
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 [56]:
```
df['CountryCode']
```

Out[56]:
```
0      ABW
1      AFG
2      AGO
3      ALB
4      ARE
      ...
190    YEM
191    ZAF
192    COD
193    ZMB
194    ZWE
Name: CountryCode, Length: 195, dtype: object
```

In [57]:
```
df['CountryName','CountryCode']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\AppData\Roaming\Python\Python312\site-packages\pandas\core\indexes\base.p
y:3812, in Index.get_loc(self, key)
   3811 try:
-> 3812     return self._engine.get_loc(casted_key)
   3813 except KeyError as err:

File pandas/_libs/index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.PyOb
jectHashTable.get_item()

File pandas/_libs/hashtable_class_helper.pxi:7096, in pandas._libs.hashtable.PyOb
jectHashTable.get_item()

KeyError: ('CountryName', 'CountryCode')

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[57], line 1
----> 1 df['CountryName','CountryCode']

File ~\AppData\Roaming\Python\Python312\site-packages\pandas\core\frame.py:4107,
in DataFrame.__getitem__(self, key)
   4105 if self.columns.nlevels > 1:
   4106     return self._getitem_multilevel(key)
-> 4107 indexer = self.columns.get_loc(key)
   4108 if is_integer(indexer):
   4109     indexer = [indexer]

File ~\AppData\Roaming\Python\Python312\site-packages\pandas\core\indexes\base.p
y:3819, in Index.get_loc(self, key)
   3814     if isinstance(casted_key, slice) or (
   3815         isinstance(casted_key, abc.Iterable)
   3816         and any(isinstance(x, slice) for x in casted_key)
   3817     ):
   3818         raise InvalidIndexError(key)
-> 3819     raise KeyError(key) from err
   3820 except TypeError:
   3821     # If we have a listlike key, _check_indexing_error will raise
   3822     #  InvalidIndexError. Otherwise we fall through and re-raise
   3823     #  the TypeError.
   3824     self._check_indexing_error(key)

KeyError: ('CountryName', 'CountryCode')
```

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

In [ ]:
```
df[['CountryName','CountryCode','IncomeGroup']]
```

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

In [ ]:
```
print(len(df))
print(len(df_cat))
```

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

In [ ]:
```
print(len(df_cat.columns))
```

In [ ]:
```
print((df.columns))
```

In [ ]:
```
print(df_cat.columns)
```

In [ ]:
```
df_cat.describe()
```

In [ ]:
```
df_num=df[['BirthRate','InternetUsers']]
```

In [ ]:
```
df_num
```

In [ ]:
```
df.info()
```

In [ ]:
```
df_cat.info()
```

In [ ]:
```
df_num.info()
```

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

In [ ]:
```
df.describe().transpose() # rows are converted to columns and columns are conver
```

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

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

In [ ]:
```
df.columns=['a','b','c','d','e'] # rename the columns
```

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

In [ ]:
```
df.head(1)
```

In [ ]:
```
df.columns=['CountryName','CountryCode','BirthRate','InternetUsers','IncomeGroup
```

In [ ]:
```
df.head(1)
```

In [ ]:
```
df[['CountryName','CountryCode','BirthRate','InternetUsers']][4:8] # subset
```

In [ ]:
```
df[4:8][['CountryName','CountryCode','BirthRate','InternetUsers']]
```

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

In [ ]:
```
df. BirthRate*df.InternetUsers # multiply two columns
```

In [ ]:
```
df.head(2)
```

In [ ]:
```
df['newcolumn']=df. BirthRate*df.InternetUsers
```

In [ ]:
```
df.head(5)
```

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

In [ ]:
```
df=df.drop('newcolumn',axis=1) # axis =1 means it deletes the column and delete
```

In [ ]:
```
df.head(1)
```

In [ ]:
```
df=df.drop('newcolumn',axis=1) # column already deleted
```

In [ ]:
```
df
```

In [ ]:
```
df.InternetUsers<2 #
```

In [ ]:
```
df[df.InternetUsers<2]
```

In [ ]:
```
len(df[df.InternetUsers<2])
```

In [ ]:
```
df.BirthRate>40
```

In [ ]:
```
df[df.BirthRate>40]
```

In [ ]:
```
len(df[df.BirthRate>40])
```

In [ ]:
```
low_InternetUsers_Country=df.InternetUsers<2
low_InternetUsers_Country
```

In [ ]:
```
high_birth_rate=df.BirthRate>40
high_birth_rate
```

In [ ]:
```
low_InternetUsers_Country & high_birth_rate
```

In [ ]:
```
df[df.InternetUsers<2] & df[df.BirthRate>40]
```

In [ ]:
```
df.InternetUsers<2 & df.BirthRate>40
```

In [ ]:
```
Filter=df.InternetUsers < 2
Filter2=df.BirthRate>40
```

In [ ]:
```
df[Filter & Filter2] # internetusers<2 and birthrate>40
```

In [ ]:
```
df[low_InternetUsers_Country & high_birth_rate]
```

In [ ]:
```
df_num
```

In [ ]:
```
df_cat
```

In [ ]:
```
  df[df.IncomeGroup=='High income']
```

In [ ]:
```
 df.IncomeGroup=='High income'
```

In [ ]:
```
df[df.IncomeGroup=='High income']
```

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

In [ ]:
```
df.IncomeGroup.unique()
```

In [59]:
```
df.IncomeGroup.nunique()
```

Out[59]: 4

we analysis python dataframe or dataset

In [61]:
```
import matplotlib.pyplot as plt # visualization
import seaborn as sns # stats visualization ,advanced visualization


#%matplotlib inline #plot the graph in the line
plt.rcParams['figure.figsize'] = (6,2) #width=6,height=2


import warnings
warnings.filterwarnings('ignore') # whenver os will update.ignore the os  error
```

In [62]:
```
df.head()
```

Out[62]:

	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 [63]:
```
df.columns
```

Out[63]:
```
Index(['CountryName', 'CountryCode', 'BirthRate', 'InternetUsers',
       'IncomeGroup'],
      dtype='object')
```

In [64]:
```
df['InternetUsers']
```

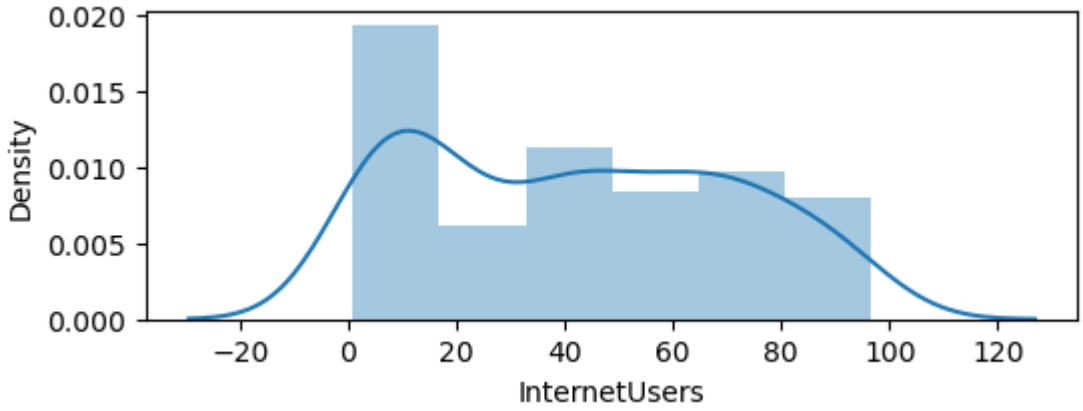
Out[64]:
```
0      78.9
1       5.9
2      19.1
3      57.2
4      88.0
       ...
190    20.0
191    46.5
192     2.2
193    15.4
194    18.5
Name: InternetUsers, Length: 195, dtype: float64
```

In [65]:
```
vis1=plt.distplot(df["InternetUsers"])
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[65], line 1
----> 1 vis1=plt.distplot(df["InternetUsers"])

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

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

In [66]:
```
vis1 = sns.distplot(df["InternetUsers"])
plt.show(vis1)
```

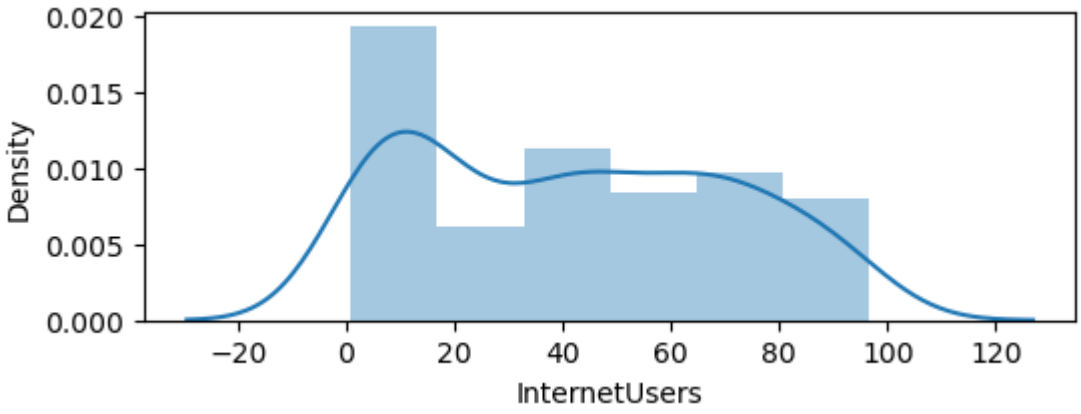


In [67]:
```
df['InternetUsers']
```

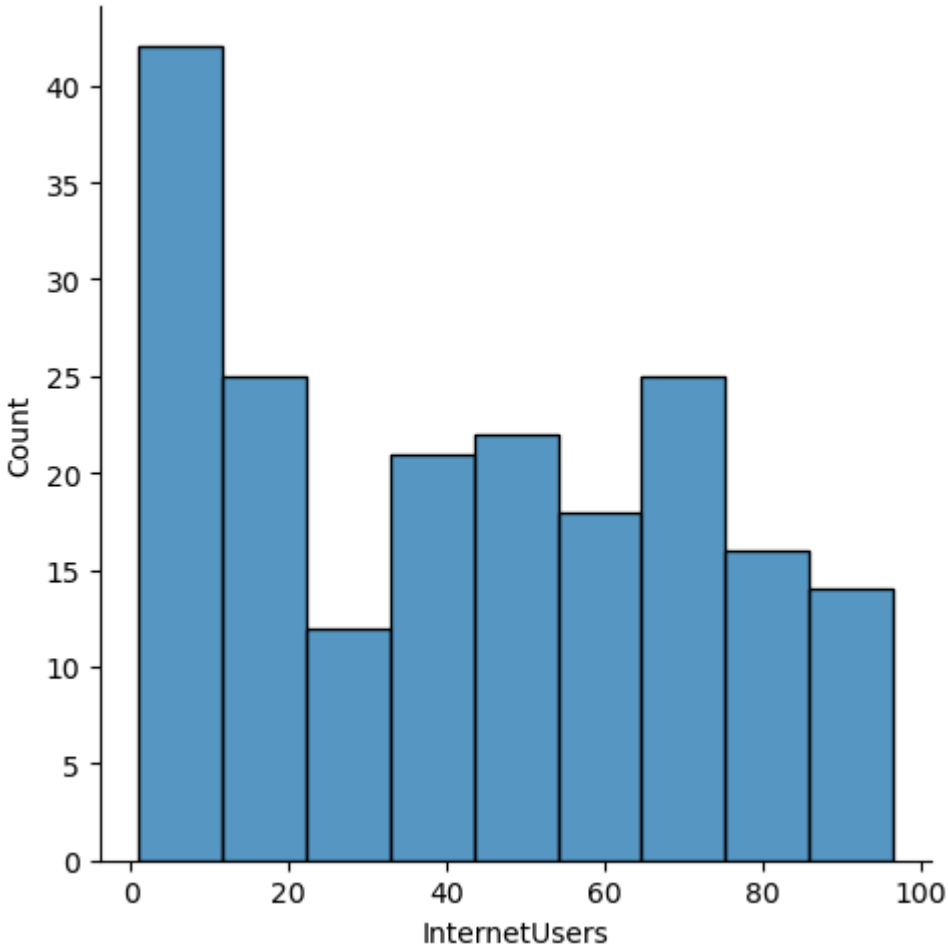
Out[67]:
```
0      78.9
1       5.9
2      19.1
3      57.2
4      88.0
       ...
190    20.0
191    46.5
192     2.2
193    15.4
194    18.5
Name: InternetUsers, Length: 195, dtype: float64
```

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

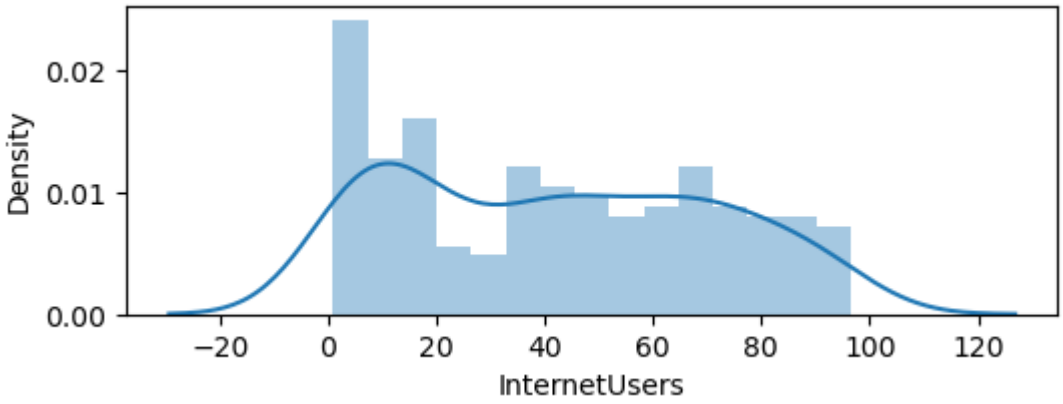
In [68]:
```
vis1=sns.distplot(df['InternetUsers']) #distplot means distributed line
# univariate analysis--plot the graph using 1 variable is called univariate anal
```



In [70]:
```
vis2=sns.displot(df['InternetUsers'])
```

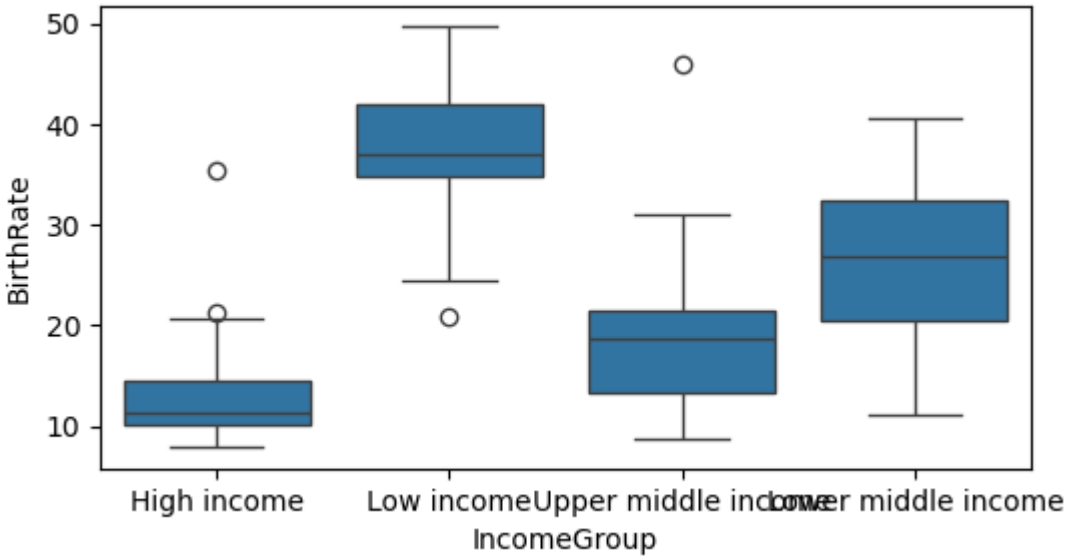


In [71]:
```
vis3=sns.distplot(df['InternetUsers'],bins=15)
```



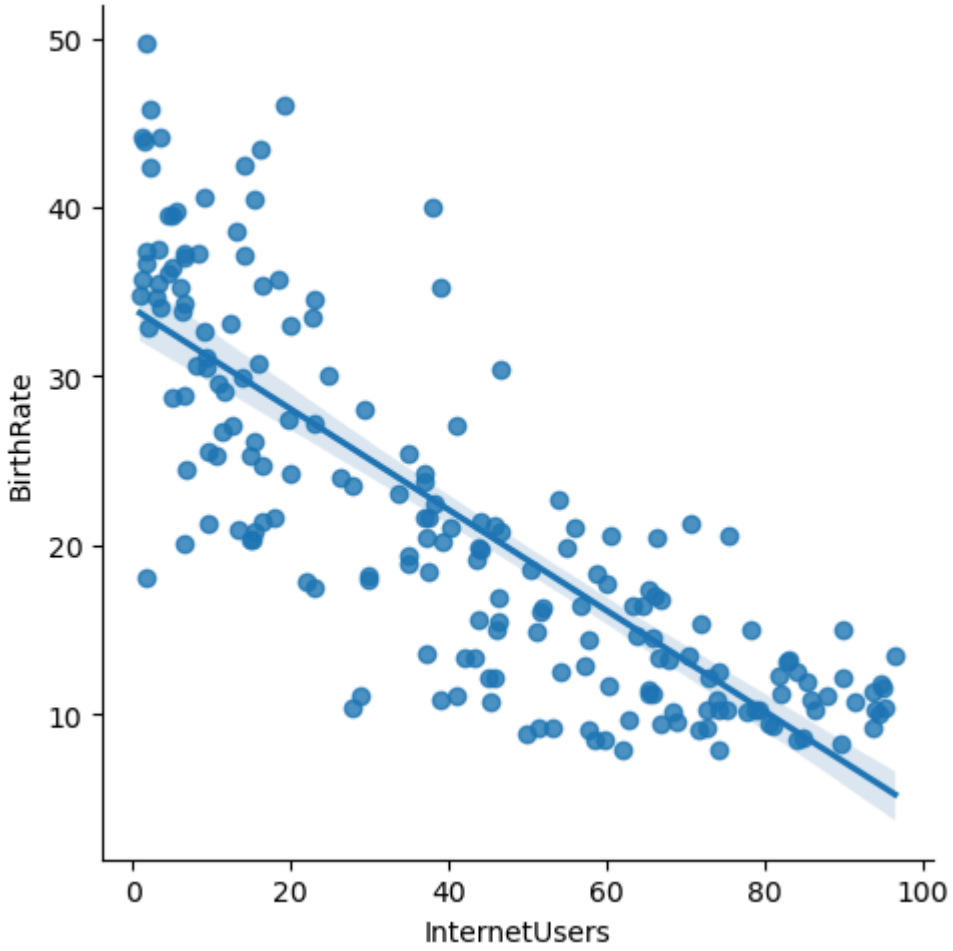
In [78]:
```
plt.rcParams['figure.figsize']=6,3
```

In [80]:
```
vis4=sns.boxplot(data=df,x='IncomeGroup',y='BirthRate')
# Byvariant means plot the graph using two variables is called Bivariant analysi
```



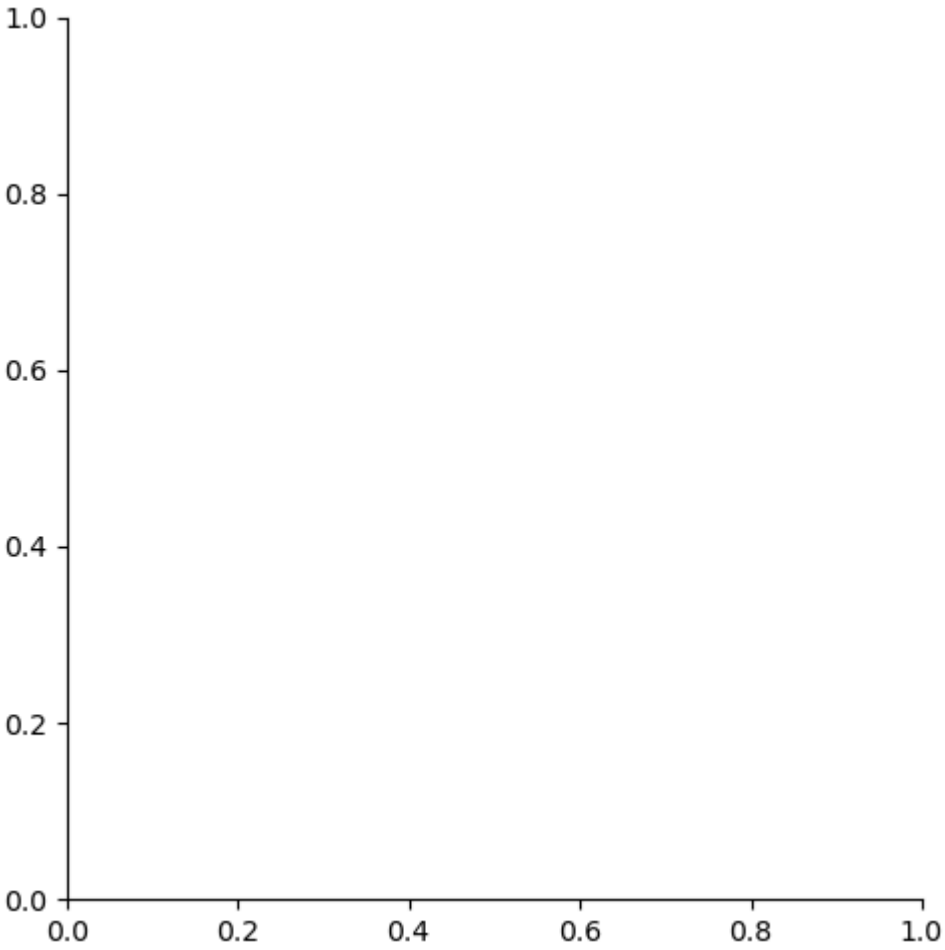
In [81]:
```
# outlier-which doesnot obey the common behaviour
#in statistics outlier is the datapoint which is very far from other objects/obs
# outlier is also called as Anomaly Detection
```

In [84]:
```
vis5=sns.lmplot(data=df,x='InternetUsers',y='BirthRate')#lmplot- linear model pl
```



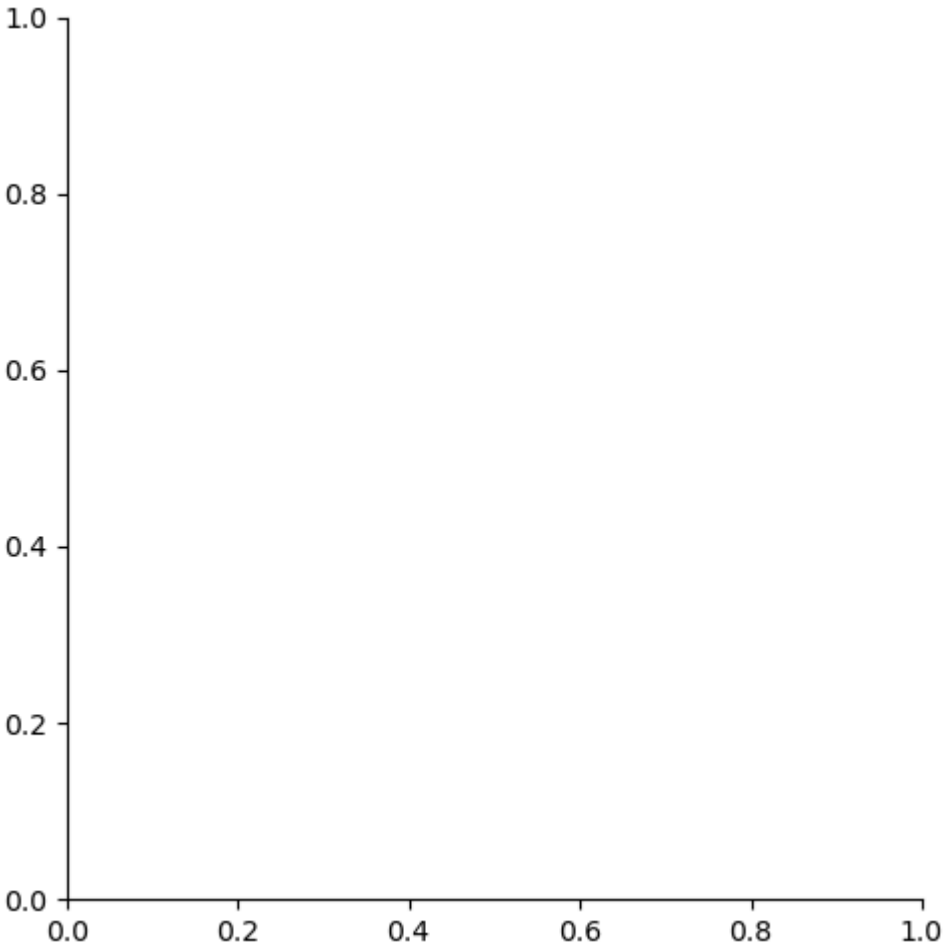
In [88]:
```
vs=sns.lmplot(data=df,x='BirthRate',y='IncomeGroup')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[88], line 1
----> 1 vs=sns.lmplot(data=df,x='BirthRate',y='IncomeGroup')

File ~\anaconda3\Lib\site-packages\seaborn\regression.py:640, in lmplot(data, x,
y, hue, col, row, palette, col_wrap, height, aspect, markers, sharex, sharey, hue
_order, col_order, row_order, legend, legend_out, x_estimator, x_bins, x_ci, scat
ter, fit_reg, ci, n_boot, units, seed, order, logistic, lowess, robust, logx, x_p
artial, y_partial, truncate, x_jitter, y_jitter, scatter_kws, line_kws, facet_kw
s)
    637     ax.update_datalim(xys, updatey=False)
    638     ax.autoscale_view(scaley=False)
--> 640 facets.map_dataframe(update_datalim, x=x, y=y)
    642 # Draw the regression plot on each facet
    643 regplot_kws = dict(
    644     x_estimator=x_estimator, x_bins=x_bins, x_ci=x_ci,
    645     scatter=scatter, fit_reg=fit_reg, ci=ci, n_boot=n_boot, units=units,
   (...)
    649     scatter_kws=scatter_kws, line_kws=line_kws,
    650 )

File ~\anaconda3\Lib\site-packages\seaborn\axisgrid.py:825, in FacetGrid.map_data
frame(self, func, *args, **kwargs)
    822     kwargs["data"] = data_ijk
    824     # Draw the plot
--> 825     self._facet_plot(func, ax, args, kwargs)
    827 # For axis labels, prefer to use positional args for backcompat
    828 # but also extract the x/y kwargs and use if no corresponding arg
    829 axis_labels = [kwargs.get("x", None), kwargs.get("y", None)]

File ~\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854, in FacetGrid._facet_p
lot(self, func, ax, plot_args, plot_kwargs)
    852     plot_args = []
    853     plot_kwargs["ax"] = ax
--> 854 func(*plot_args, **plot_kwargs)
    856 # Sort out the supporting information
    857 self._update_legend_data(ax)

File ~\anaconda3\Lib\site-packages\seaborn\regression.py:636, in lmplot.<locals>.
update_datalim(data, x, y, ax, **kws)
    635 def update_datalim(data, x, y, ax, **kws):
--> 636     xys = data[[x, y]].to_numpy().astype(float)
    637     ax.update_datalim(xys, updatey=False)
    638     ax.autoscale_view(scaley=False)

ValueError: could not convert string to float: 'High income'
```



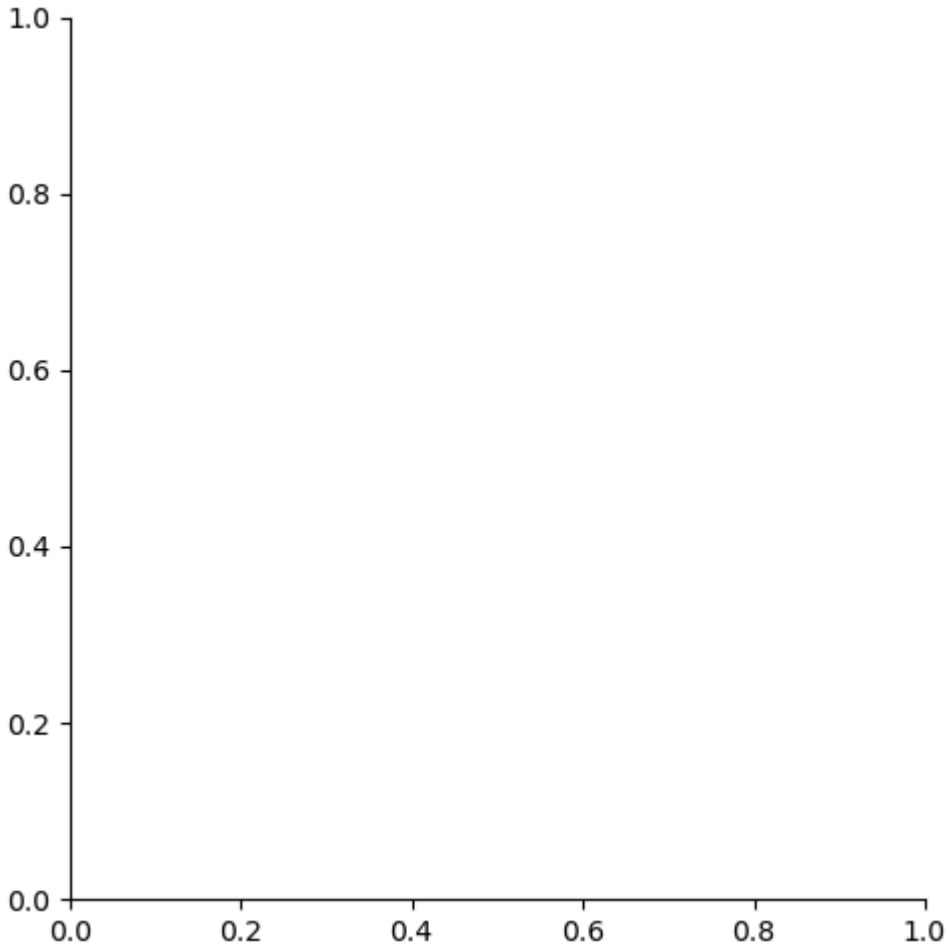
In [89]:
```
vs=sns.lmplot(data=df,x='CountryCode',y='IncomeGroup')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[89], line 1
----> 1 vs=sns.lmplot(data=df,x='CountryCode',y='IncomeGroup')

File ~\anaconda3\Lib\site-packages\seaborn\regression.py:640, in lmplot(data, x,
y, hue, col, row, palette, col_wrap, height, aspect, markers, sharex, sharey, hue
_order, col_order, row_order, legend, legend_out, x_estimator, x_bins, x_ci, scat
ter, fit_reg, ci, n_boot, units, seed, order, logistic, lowess, robust, logx, x_p
artial, y_partial, truncate, x_jitter, y_jitter, scatter_kws, line_kws, facet_kw
s)
    637     ax.update_datalim(xys, updatey=False)
    638     ax.autoscale_view(scaley=False)
--> 640 facets.map_dataframe(update_datalim, x=x, y=y)
    642 # Draw the regression plot on each facet
    643 regplot_kws = dict(
    644     x_estimator=x_estimator, x_bins=x_bins, x_ci=x_ci,
    645     scatter=scatter, fit_reg=fit_reg, ci=ci, n_boot=n_boot, units=units,
   (...)
    649     scatter_kws=scatter_kws, line_kws=line_kws,
    650 )

File ~\anaconda3\Lib\site-packages\seaborn\axisgrid.py:825, in FacetGrid.map_data
frame(self, func, *args, **kwargs)
    822     kwargs["data"] = data_ijk
    824     # Draw the plot
--> 825     self._facet_plot(func, ax, args, kwargs)
    827 # For axis labels, prefer to use positional args for backcompat
    828 # but also extract the x/y kwargs and use if no corresponding arg
    829 axis_labels = [kwargs.get("x", None), kwargs.get("y", None)]

File ~\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854, in FacetGrid._facet_p
lot(self, func, ax, plot_args, plot_kwargs)
    852     plot_args = []
    853     plot_kwargs["ax"] = ax
--> 854 func(*plot_args, **plot_kwargs)
    856 # Sort out the supporting information
    857 self._update_legend_data(ax)

File ~\anaconda3\Lib\site-packages\seaborn\regression.py:636, in lmplot.<locals>.
update_datalim(data, x, y, ax, **kws)
    635 def update_datalim(data, x, y, ax, **kws):
--> 636     xys = data[[x, y]].to_numpy().astype(float)
    637     ax.update_datalim(xys, updatey=False)
    638     ax.autoscale_view(scaley=False)

ValueError: could not convert string to float: 'ABW'
```

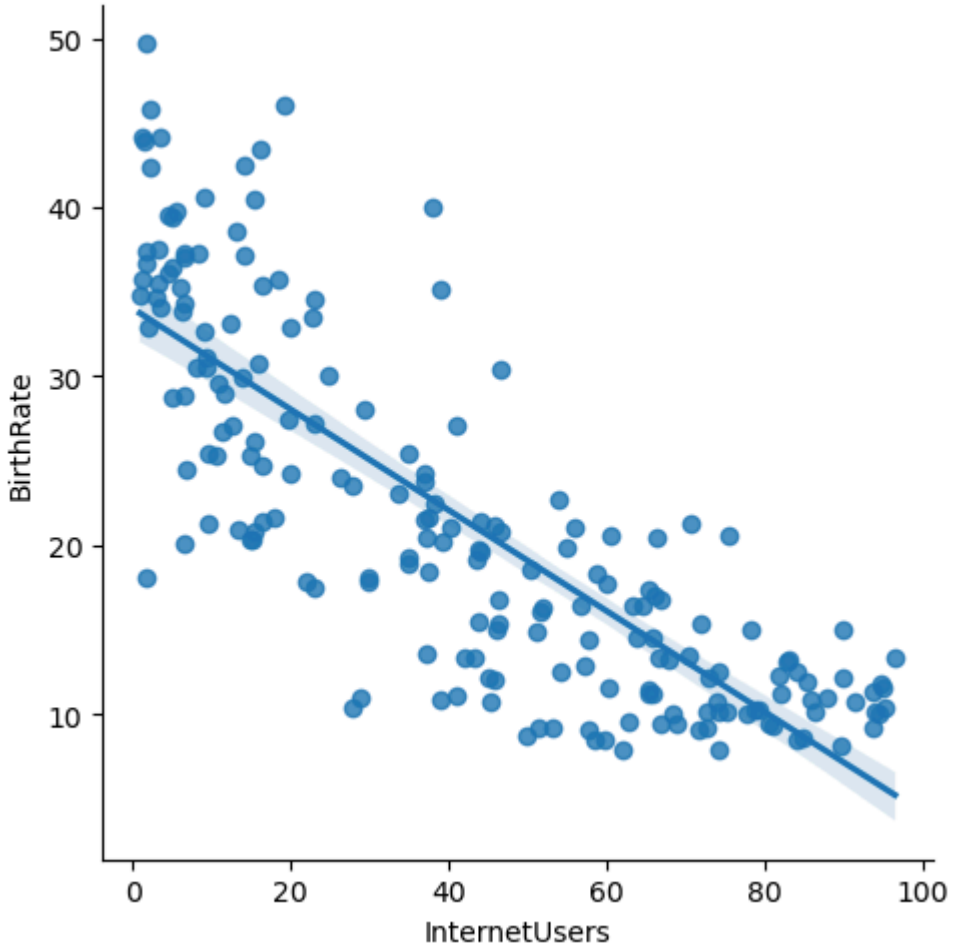


In [90]:
```
vs=sns.lmplot(data=df,x='CountryName',y='IncomeGroup')
```

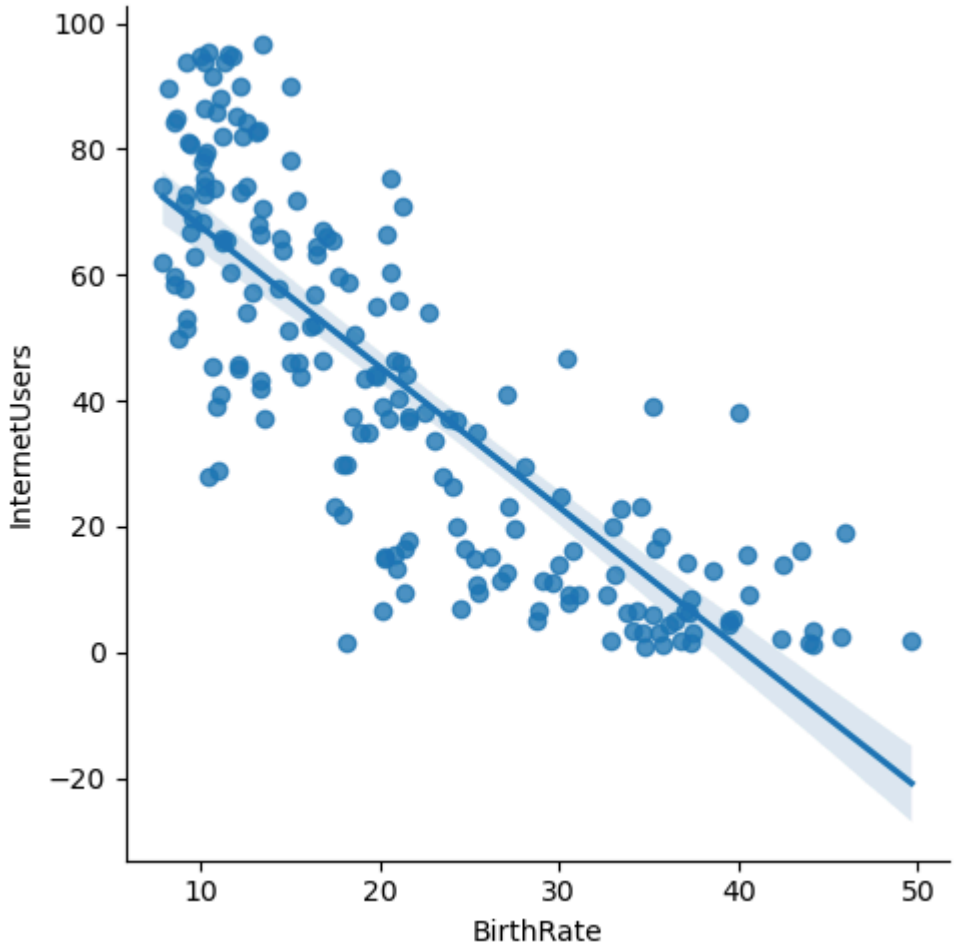
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[90], line 1
----> 1 vs=sns.lmplot(data=df,x='CountryName',y='IncomeGroup')

File ~\anaconda3\Lib\site-packages\seaborn\regression.py:640, in lmplot(data, x,
y, hue, col, row, palette, col_wrap, height, aspect, markers, sharex, sharey, hue
_order, col_order, row_order, legend, legend_out, x_estimator, x_bins, x_ci, scat
ter, fit_reg, ci, n_boot, units, seed, order, logistic, lowess, robust, logx, x_p
artial, y_partial, truncate, x_jitter, y_jitter, scatter_kws, line_kws, facet_kw
s)
    637     ax.update_datalim(xys, updatey=False)
    638     ax.autoscale_view(scaley=False)
--> 640 facets.map_dataframe(update_datalim, x=x, y=y)
    642 # Draw the regression plot on each facet
    643 regplot_kws = dict(
    644     x_estimator=x_estimator, x_bins=x_bins, x_ci=x_ci,
    645     scatter=scatter, fit_reg=fit_reg, ci=ci, n_boot=n_boot, units=units,
   (...)
    649     scatter_kws=scatter_kws, line_kws=line_kws,
    650 )

File ~\anaconda3\Lib\site-packages\seaborn\axisgrid.py:825, in FacetGrid.map_data
frame(self, func, *args, **kwargs)
    822     kwargs["data"] = data_ijk
    824     # Draw the plot
--> 825     self._facet_plot(func, ax, args, kwargs)
    827 # For axis labels, prefer to use positional args for backcompat
    828 # but also extract the x/y kwargs and use if no corresponding arg
    829 axis_labels = [kwargs.get("x", None), kwargs.get("y", None)]

File ~\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854, in FacetGrid._facet_p
lot(self, func, ax, plot_args, plot_kwargs)
    852     plot_args = []
    853     plot_kwargs["ax"] = ax
--> 854 func(*plot_args, **plot_kwargs)
    856 # Sort out the supporting information
    857 self._update_legend_data(ax)

File ~\anaconda3\Lib\site-packages\seaborn\regression.py:636, in lmplot.<locals>.
update_datalim(data, x, y, ax, **kws)
    635 def update_datalim(data, x, y, ax, **kws):
--> 636     xys = data[[x, y]].to_numpy().astype(float)
    637     ax.update_datalim(xys, updatey=False)
    638     ax.autoscale_view(scaley=False)

ValueError: could not convert string to float: 'Aruba'
```



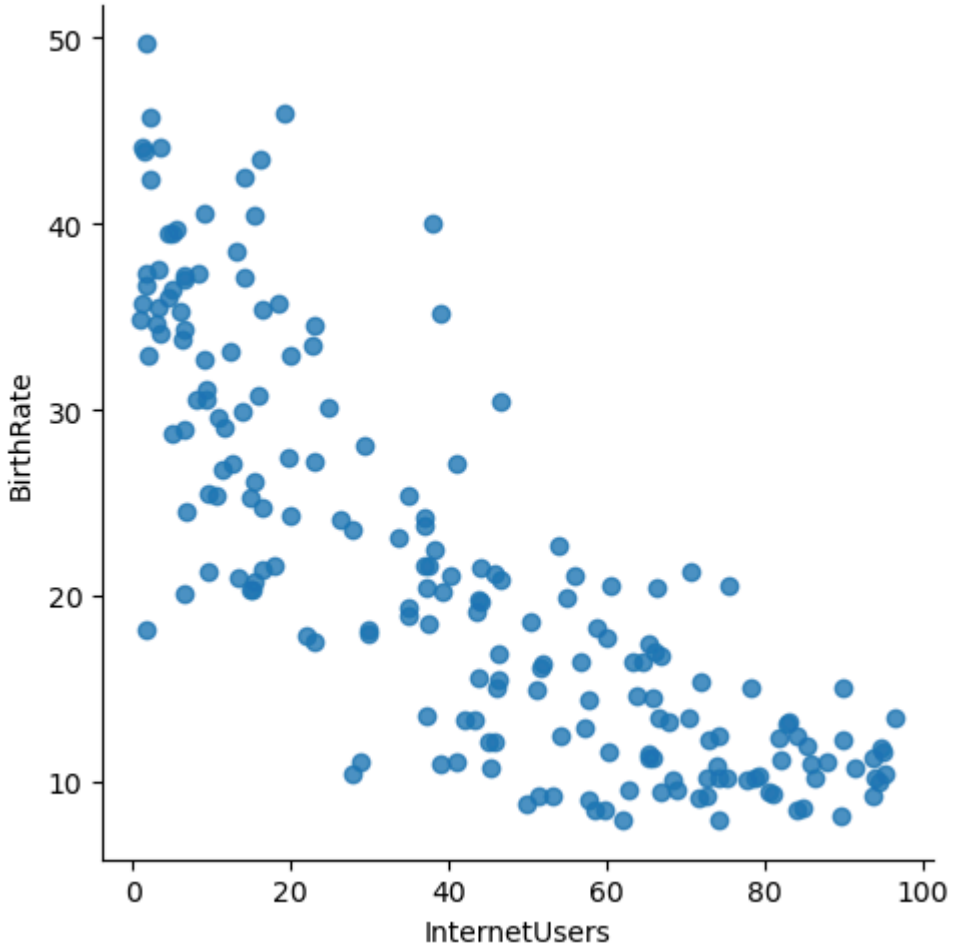
In [92]:
```
vis5=sns.lmplot(data=df,x='InternetUsers',y='BirthRate')
```



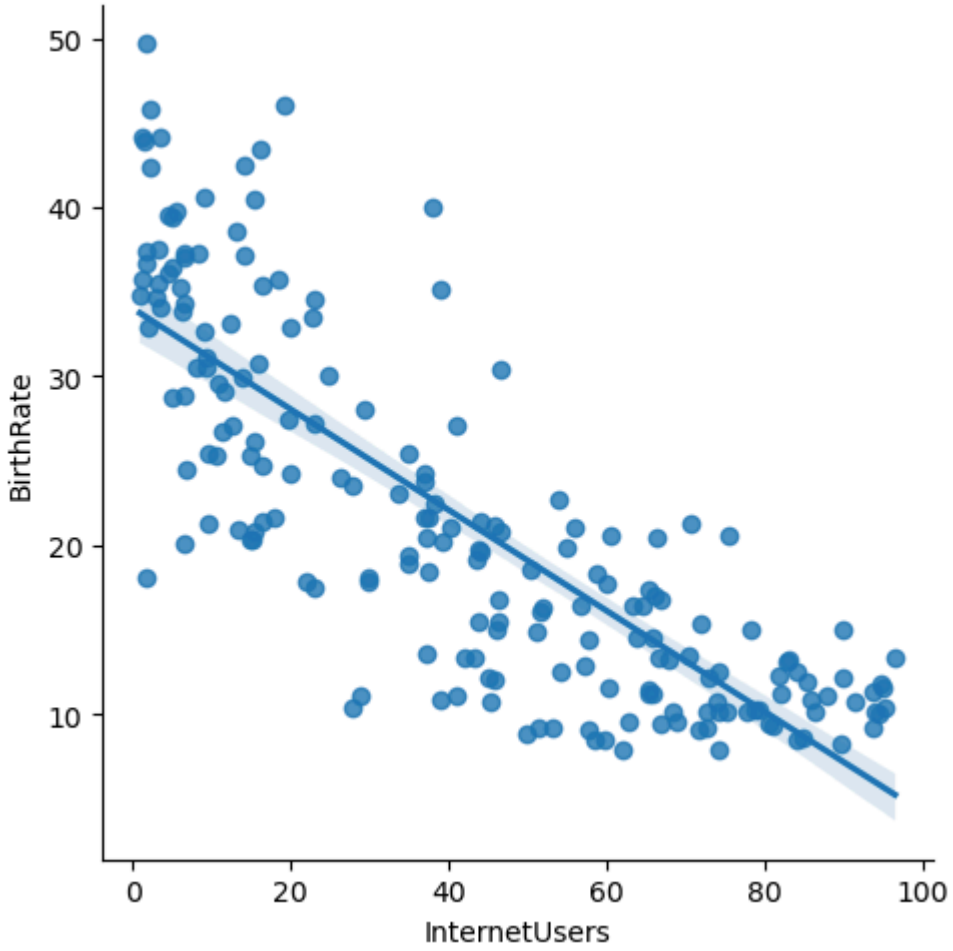
In [93]:
```
vis5=sns.lmplot(data=df,x='BirthRate',y='InternetUsers')
```



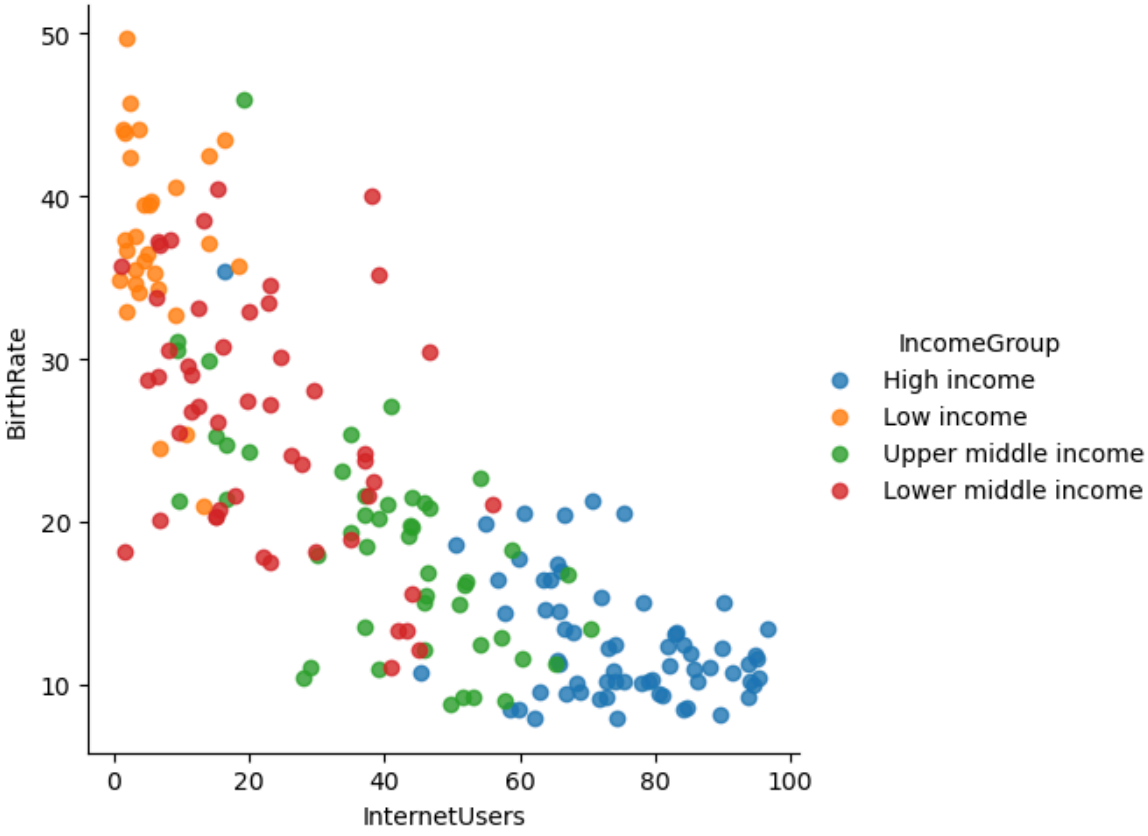
In [96]:
```
vis5=sns.lmplot(data=df,x='InternetUsers',y='BirthRate',fit_reg=False) # Regress
```



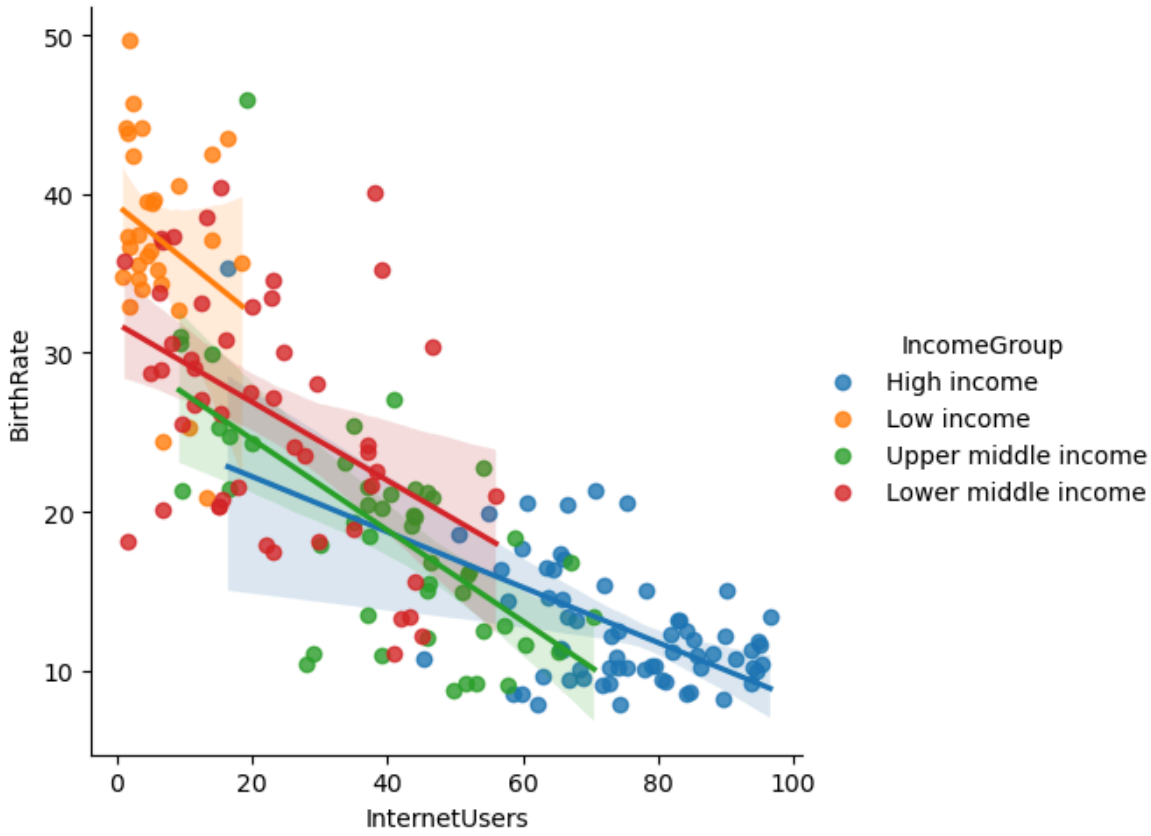
In [97]:
```
vis5=sns.lmplot(data=df,x='InternetUsers',y='BirthRate',fit_reg=True)
```



In [98]:
```
vis8=sns.lmplot(data=df,x='InternetUsers',y='BirthRate',fit_reg=False,hue='Incom
```

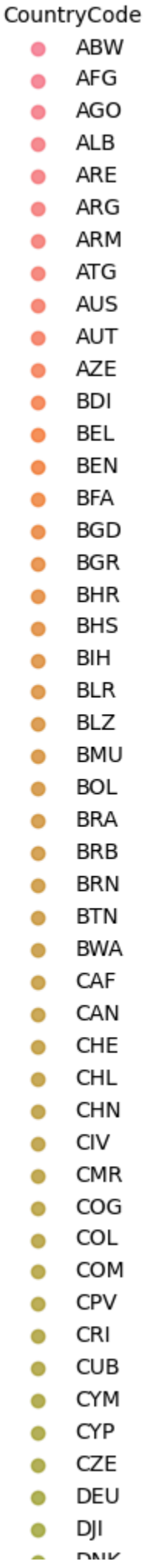


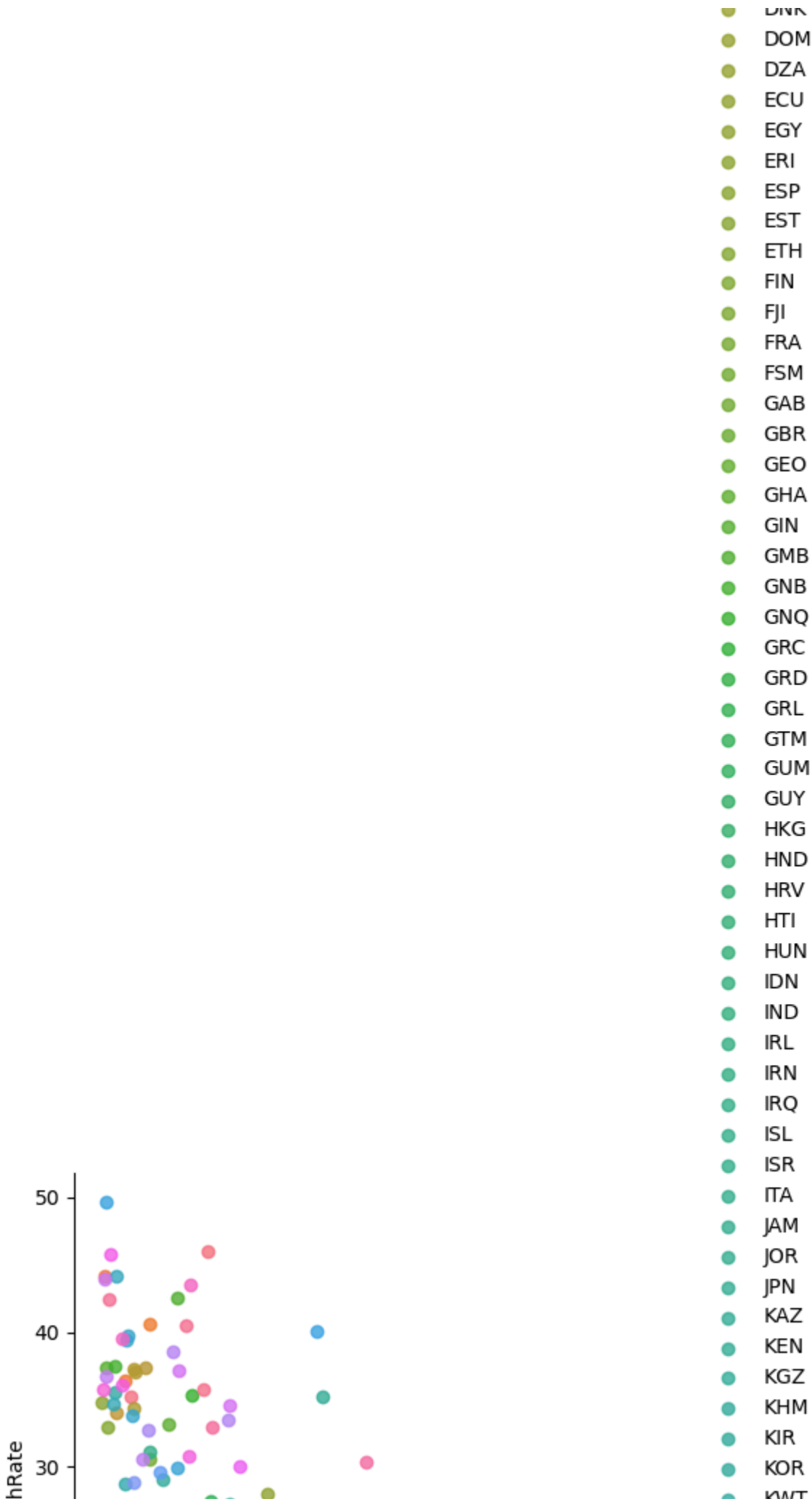
In [100…
```
vis8=sns.lmplot(data=df,x='InternetUsers',y='BirthRate',fit_reg=True,hue='Income
```

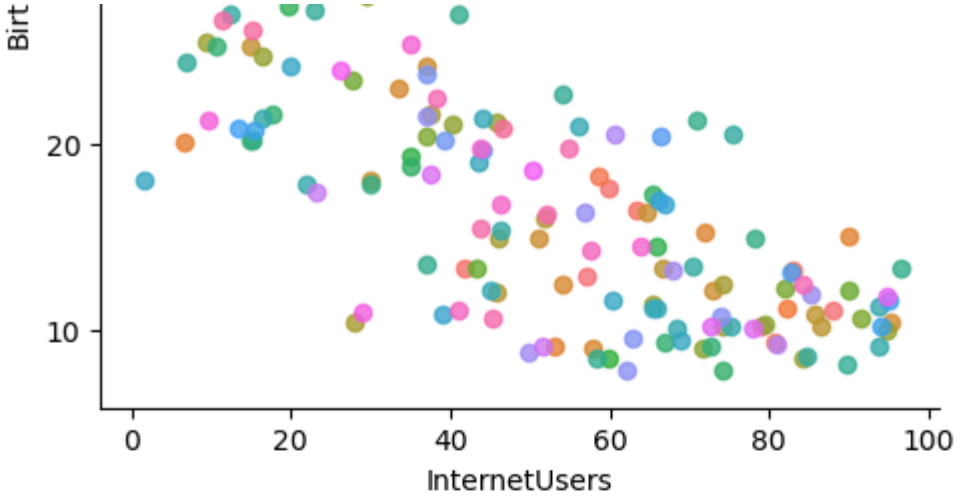


In [106…

```
vis8=sns.lmplot(data=df,x='InternetUsers',y='BirthRate',fit_reg=False,hue='Count
```


CountryCode
ABW
AFG
AGO
ALB
ARE
ARG
ARM
ATG
AUS
AUT
AZE
BDI
BEL
BEN
BFA
BGD
BGR
BHR
BHS
BIH
BLR
BLZ
BMU
BOL
BRA
BRB
BRN
BTN
BWA
CAF
CAN
CHE
CHL
CHN
CIV
CMR
COG
COL
COM
CPV
CRI
CUB
CYM
CYP
CZE
DEU
DJI


DNK
DOM
DZA
ECU
EGY
ERI
ESP
EST
ETH
FIN
FJI
FRA
FSM
GAB
GBR
GEO
GHA
GIN
GMB
GNB
GNQ
GRC
GRD
GRL
GTM
GUM
GUY
HKG
HND
HRV
HTI
HUN
IDN
IND
IRL
IRN
IRQ
ISL
ISR
ITA
JAM
JOR
JPN
KAZ
KEN
KGZ
KHM
KIR
KOR
KWT
50
40
30
hRate



KWT
LAO
LBN
LBR
LBY
LCA
LIE
LKA
LSO
LTU
LUX
LVA
MAC
MAR
MDA
MDG
MDV
MEX
MKD
MLI
MLT
MMR
MNE
MNG
MOZ
MRT
MUS
MWI
MYS
NAM
NCL
NER
NGA
NIC
NLD
NOR
NPL
NZL
OMN
PAK
PAN
PER
PHL
PNG
POL
PRI
PRT
PRY
PYF
QAT


QAT
ROU
RUS
RWA
SAU
SDN
SEN
SGP
SLB
SLE
SLV
SOM
SRB
SSD
STP
SUR
SVK
SVN
SWE
SWZ
SYC
SYR
TCD
TGO
THA
TJK
TKM
TLS
TON
TTO
TUN
TUR
TZA
UGA
UKR
URY
USA
UZB
VCT
VEN
VIR
VNM
VUT
PSE
WSM
YEM
ZAF
COD
ZMB
ZWE

In [ ]: