World Statistics – Exploratory Data Analysis

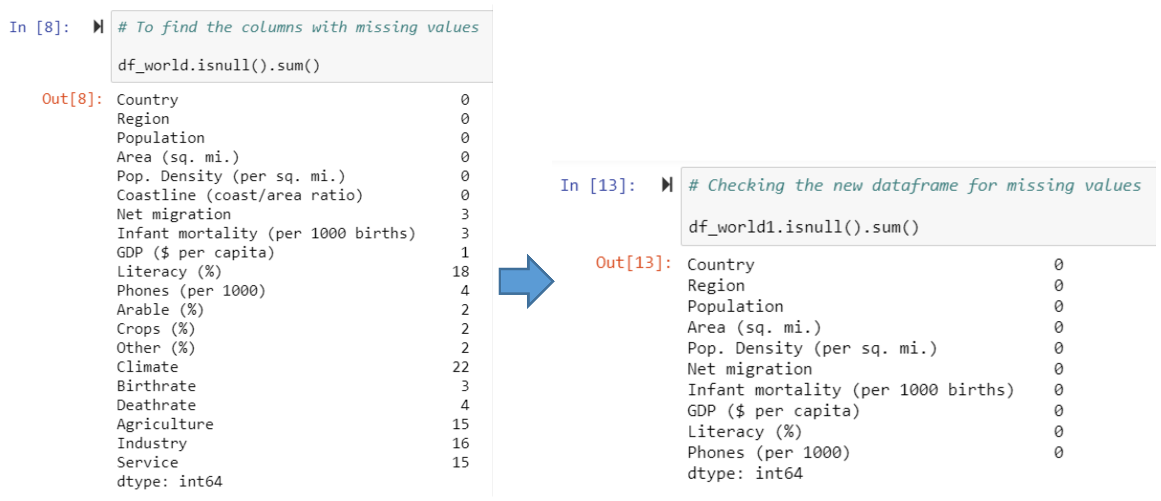
In this exercise, we will focus on making visualization more impactful through different types of charts and by working on their aesthetics and readability. The dataset contains information about different countries – such as population, area, GDP, and literacy.

Before we start visualizing data, we can have an initial look at our dataset and then clean it for visualization. The purpose of this exercise to provide a glimpse into how Python can be used to present information from a dataset in an easily digestible format.

Data Source: https://www.kaggle.com/fernandol/countries-of-the-world

Data Cleaning

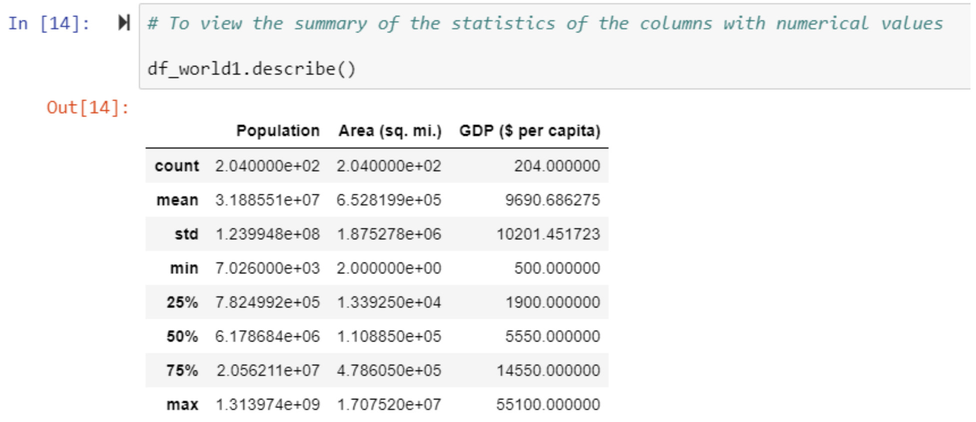
1. To deal with the missing data, we dropped the missing rows and the columns we were not focusing on.



```
In [8]:   # To find the columns with missing values

          df_world.isnull().sum()

Out[8]:   Country                              0
          Region                               0
          Population                           0
          Area (sq. mi.)                       0
          Pop. Density (per sq. mi.)           0
          Coastline (coast/area ratio)         0
          Net migration                        3
          Infant mortality (per 1000 births)   3
          GDP ($ per capita)                   1
          Literacy (%)                        18
          Phones (per 1000)                    4
          Arable (%)                           2
          Crops (%)                            2
          Other (%)                            2
          Climate                             22
          Birthrate                            3
          Deathrate                            4
          Agriculture                         15
          Industry                            16
          Service                             15
          dtype: int64
```

```
In [13]:   # Checking the new dataframe for missing values

           df_world1.isnull().sum()

Out[13]:   Country                              0
           Region                               0
           Population                           0
           Area (sq. mi.)                       0
           Pop. Density (per sq. mi.)           0
           Net migration                        0
           Infant mortality (per 1000 births)   0
           GDP ($ per capita)                   0
           Literacy (%)                         0
           Phones (per 1000)                    0
           dtype: int64
```

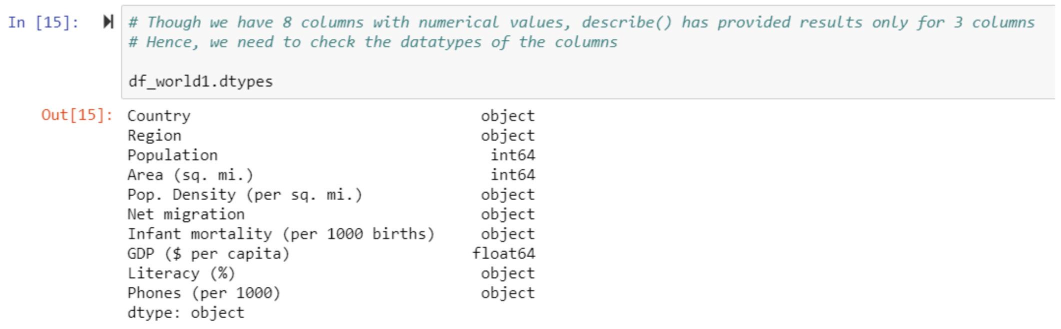
2. We viewed the summary of statistics, which had only 3 variables, but we were expecting to see other variables as well. The reason was that the other variables were not numerical.



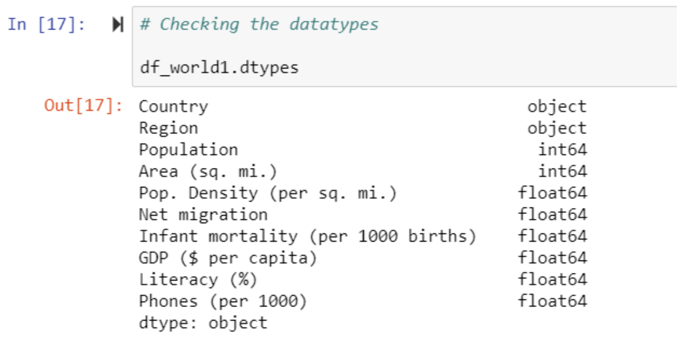
```
In [14]:   # To view the summary of the statistics of the columns with numerical values

           df_world1.describe()

Out[14]:
```

	Population	Area (sq. mi.)	GDP ($ per capita)
count	2.040000e+02	2.040000e+02	204.000000
mean	3.188551e+07	6.528199e+05	9690.686275
std	1.239948e+08	1.875278e+06	10201.451723
min	7.026000e+03	2.000000e+00	500.000000
25%	7.824992e+05	1.339250e+04	1900.000000
50%	6.178684e+06	1.108850e+05	5550.000000
75%	2.056211e+07	4.786050e+05	14550.000000
max	1.313974e+09	1.707520e+07	55100.000000

3. Hence, we converted those variables into numerical types and checked the datatypes again.

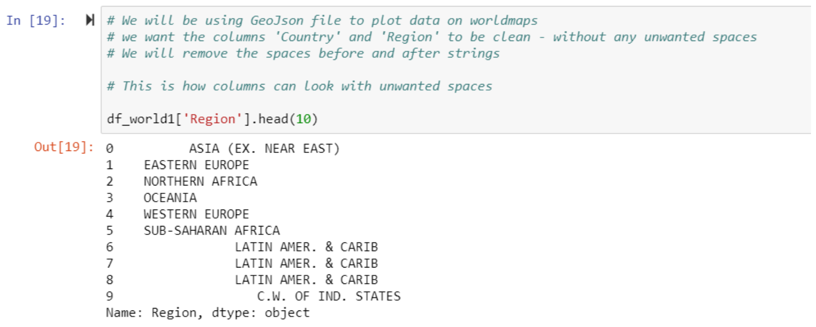


```
In [15]:  # Though we have 8 columns with numerical values, describe() has provided results only for 3 columns
          # Hence, we need to check the datatypes of the columns

          df_world1.dtypes

Out[15]:  Country                                     object
          Region                                      object
          Population                                   int64
          Area (sq. mi.)                               int64
          Pop. Density (per sq. mi.)                  object
          Net migration                               object
          Infant mortality (per 1000 births)          object
          GDP ($ per capita)                         float64
          Literacy (%)                                object
          Phones (per 1000)                           object
          dtype: object
```

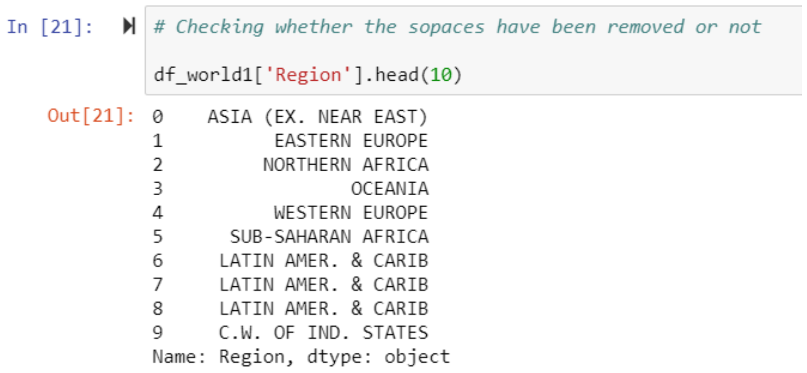


```
In [17]:  # Checking the datatypes

          df_world1.dtypes

Out[17]:  Country                                object
          Region                                 object
          Population                              int64
          Area (sq. mi.)                          int64
          Pop. Density (per sq. mi.)            float64
          Net migration                         float64
          Infant mortality (per 1000 births)    float64
          GDP ($ per capita)                    float64
          Literacy (%)                          float64
          Phones (per 1000)                     float64
          dtype: object
```

4. Further, as we needed to use GeoJSON file to plot choropleth map, the entries in 'Country' and 'Region' columns had to be clean – without any unwanted space. We removed the spaces and checked the strings again.



```
In [19]:  # We will be using GeoJson file to plot data on worldmaps
          # we want the columns 'Country' and 'Region' to be clean - without any unwanted spaces
          # We will remove the spaces before and after strings

          # This is how columns can look with unwanted spaces

          df_world1['Region'].head(10)

Out[19]:  0         ASIA (EX. NEAR EAST)
          1    EASTERN EUROPE
          2    NORTHERN AFRICA
          3    OCEANIA
          4    WESTERN EUROPE
          5    SUB-SAHARAN AFRICA
          6                LATIN AMER. & CARIB
          7                LATIN AMER. & CARIB
          8                LATIN AMER. & CARIB
          9                   C.W. OF IND. STATES
          Name: Region, dtype: object
```



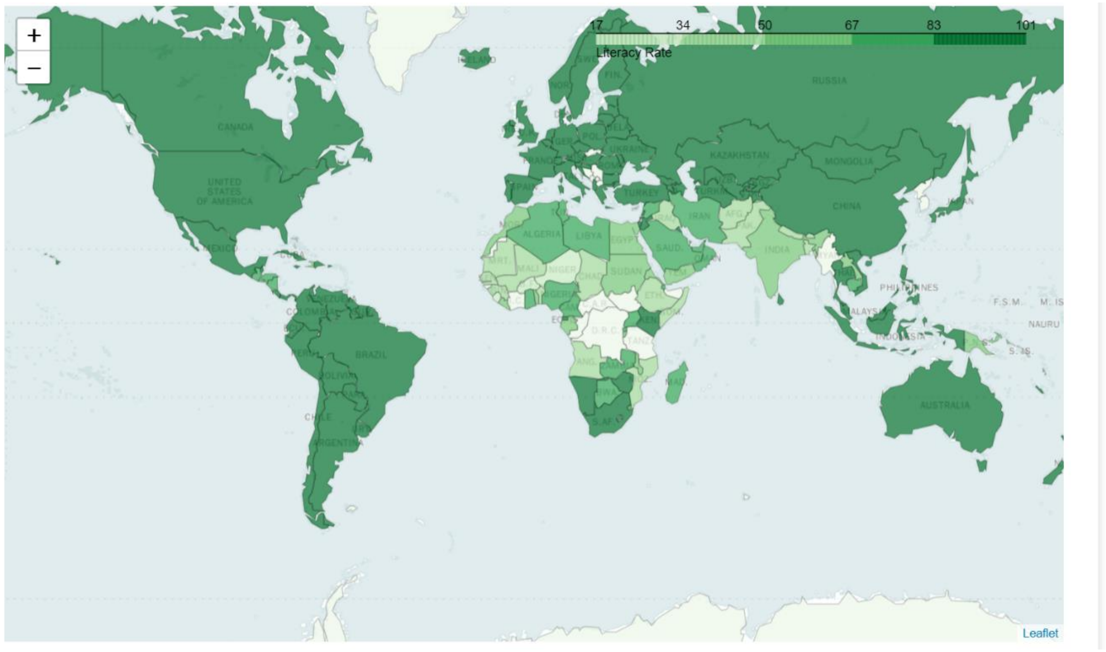
```
In [21]:  # Checking whether the sopaces have been removed or not

          df_world1['Region'].head(10)

Out[21]:  0    ASIA (EX. NEAR EAST)
          1          EASTERN EUROPE
          2         NORTHERN AFRICA
          3                 OCEANIA
          4          WESTERN EUROPE
          5      SUB-SAHARAN AFRICA
          6     LATIN AMER. & CARIB
          7     LATIN AMER. & CARIB
          8     LATIN AMER. & CARIB
          9     C.W. OF IND. STATES
          Name: Region, dtype: object
```

Exploratory Analysis

1. We used choropleth maps to show the literacy rates and infant mortality rates.

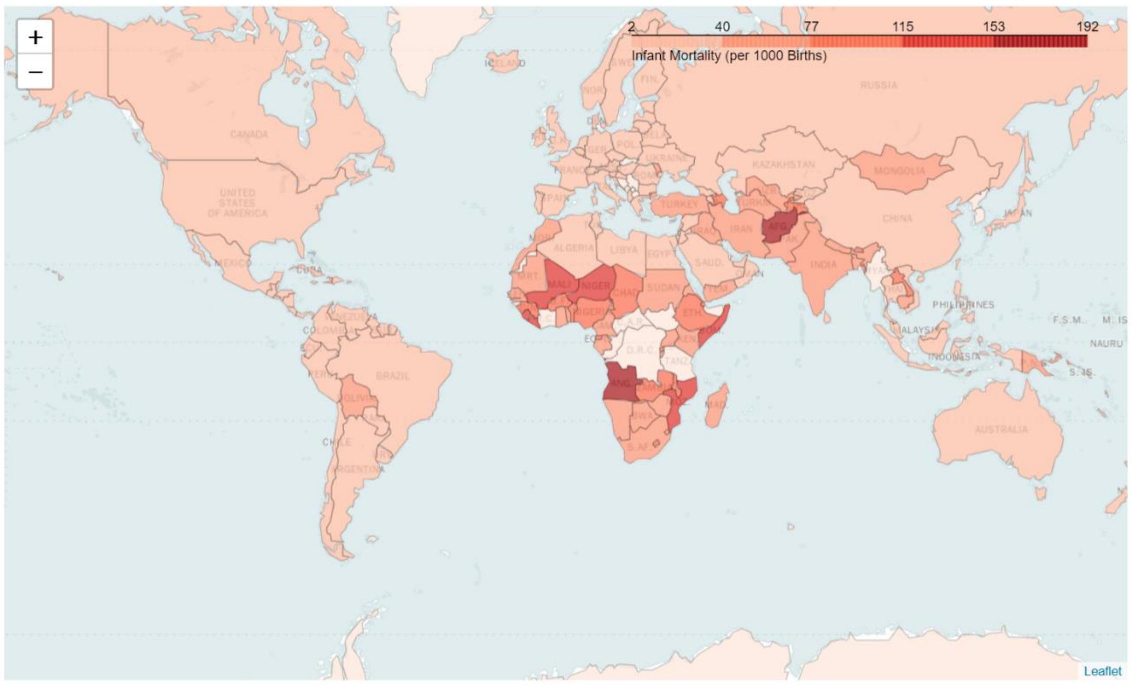
Literacy Rates

Out[23]:



Infant Mortality Rates

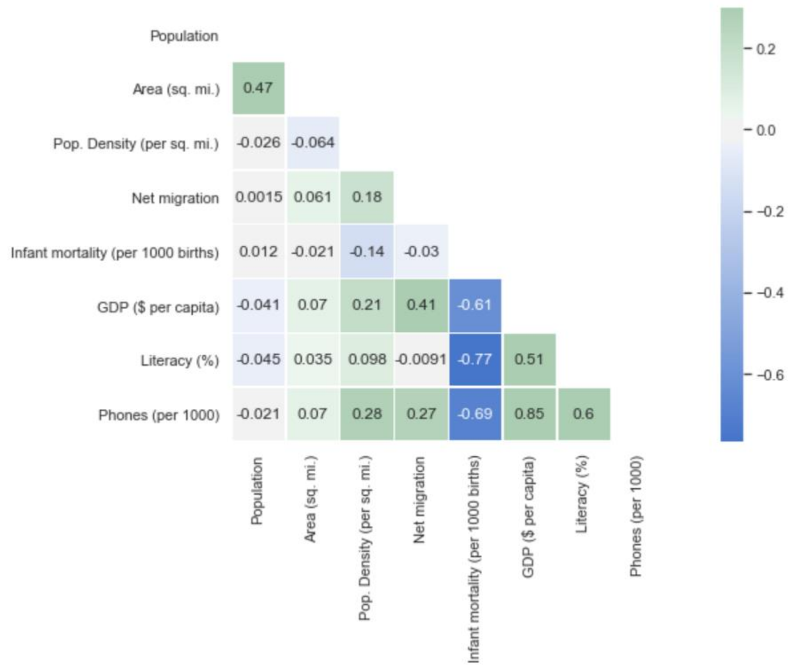
Out[24]:



Note: Some of the countries in the above two maps have a light shade, which does not depict a true value. Countries such as Ivory Coast and D.R.C (Congo) with have been removed during data cleaning or their names are not the same as in GeoJSON worldmap file.

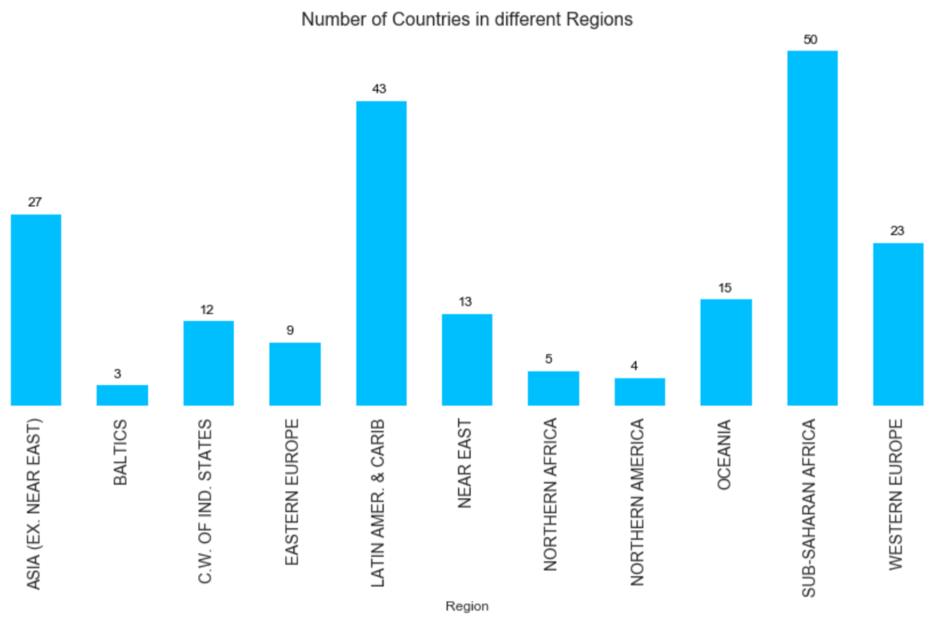
2. Correlation matrix provided us an easy view of the correlation between different numerical variables.

```
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1a87e51e908>
```

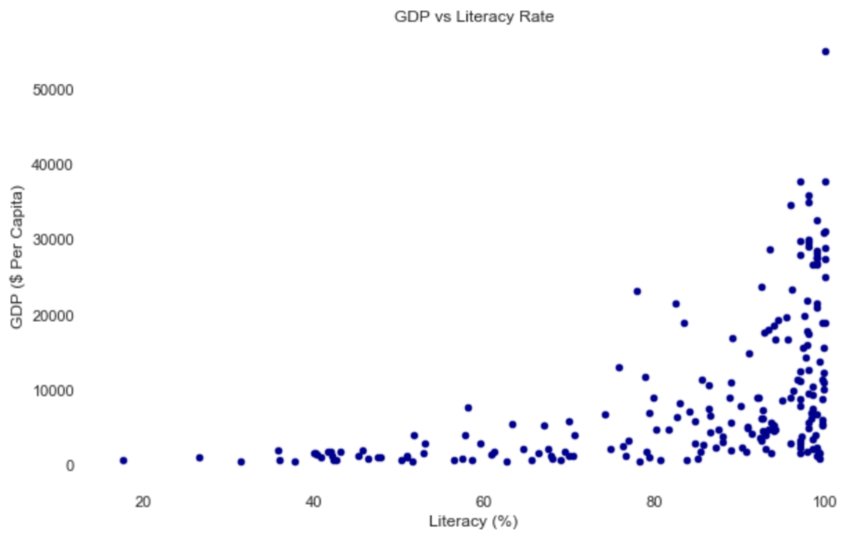


3. The bar graph we used for showing the number of countries in each region is simple and yet conveys the required information. Here, our focus was on reducing clutter and improving readability.

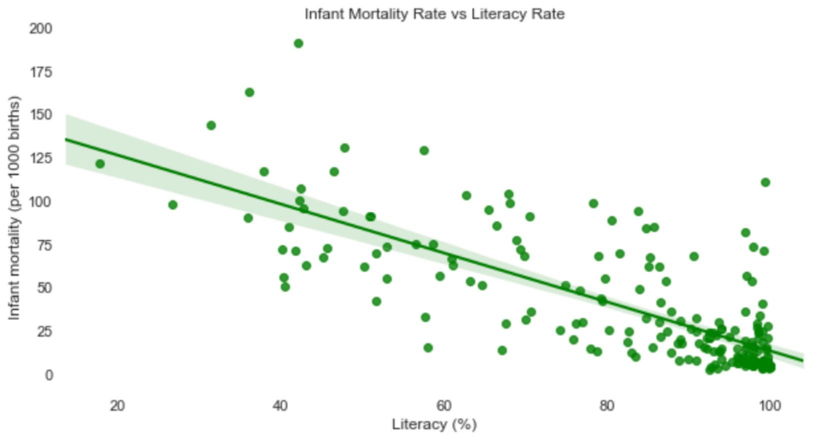
```
Out[27]: []
```



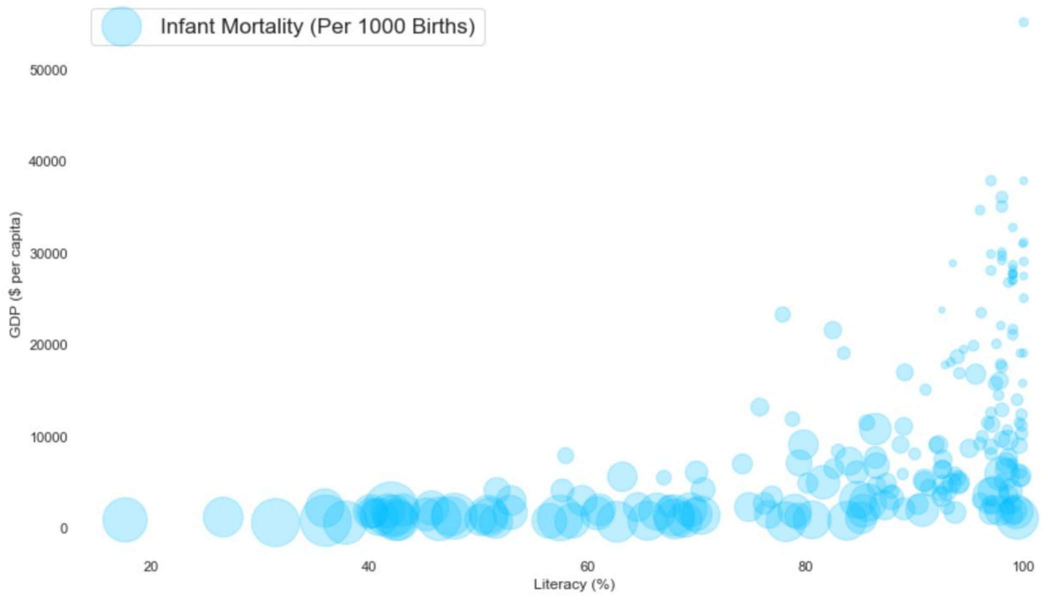
4. Finally, we created scatter plots.
   a. First one is GDP v Literacy, and it did not have a regression line



   b. Second plot is Infant Mortality v Literacy and has a regression line – showing the overall trend



   c. Third plot is a GDP vs Literacy bubble plot, with the size of the bubble indicating Infant Mortality rate. We can observe that as GDP and Literacy increases, the Infant Mortality Rate (bubble size) reduces.


Infant Mortality (Per 1000 Births)
GDP ($ per capita)
Literacy (%)