COMP5721M: Programming for Data Science

Coursework 3: Data Analysis Project

Exploratory Data Analysis and Visualisation of Global Suicide Rate

Group Members

- *Darshita Budhadev*, sc22db@leeds.ac.uk
- *Sree Lalitha Gorty*, mm22slg@leeds.ac.uk
- *Rajvamsi Chenna*, mm22rc@leeds.ac.uk
- *Venkata Bhaskar Vasista Devarakonda*, mm22vbvd@leeds.ac.uk

Project Plan

The Data (10 marks)

Suicide is a global phenomenon and as per World Health Organization, more than 700,000 people die by suicide every year, which correspomds to one person every 40 seconds (https://www.who.int/news-room/fact-sheets/detail/suicide). For this project, we used a combination of datasets to understand the different aspects underlying the prevalence of the global suicide rate. The suicide rate measures the number of suicide deaths per 100,000 in a given population. To understand the statistics of suicide rate in detail, we obtained 4 datasets from Kaggle Data Science Community, which can be accessed using the following link: (https://www.kaggle.com/datasets/twinkle0705/mental-health-and-suicide-rates?select=Facilities.csv). This dataset is licensed, but no further details regarding the acquisition of data are provided. To get a better understanding of the correlation between the population of countries and their global suicide rate in the year 2016, we obtained the 2016 world population data sheet produced by the Population Reference Bureau (PRB). The PRB data is highly accurate and could be acquired from (https://www.prb.org/wp-content/uploads/2016/08/prb-wpds2016.pdf). All these datasets will help us to investigate the changes in the global suicide rate over the 4 years and visualize the trend of the same in different age groups and genders. Furthermore, taking into account the mental health facilities and resources dataset, this will additionally shed light on integrating various measures for each country that requires improvement, for suicide prevention.

Age-standardised Suicide Rates Dataset

This dataset gives the change in global suicide rate over 4 years for a large number of countries.

- Country
  - 183 unique values of country names in the world.
  - *String object*
- Sex
  - 3 values: Male, Female and Both Sexes.

    - *Categorical variable*
- Year
    - 4 different year columns present: 2016, 2015, 2010 and 2000.
    - *All the 4 columns contain float variables*

Crude Suicide Rates Dataset

This dataset provides insights about the pattern of suicide rate per 100K for 8 different age groups and 3 categories of genders for all countries.A population column was also added to this dataset to undesrtand its interconnection with the global suicide rate.

- Country
    - 183 country inputs across the world.
    - *String object*
- Sex
    - 3 dfferent types: Male, Female and Both Sexes.
    - *Categorical variable*
- Age bands
    - Data for 8 different age groups in the year 2016, namely: 80_above, 70to79, 60to69,50to59, 40-49, 30-39, 20-29 and 10-19.
    - *All of the 8 columns have float variables*
- Population
    - Total population of all 183 countries in millions for the year 2016.
    - *Float variable*

Health Care Facility Dataset

This dataset holds the number of all Health care facility units available per every 100K population within each country across the world. The health care facilities are categorized into 4 domains based on the type of service they provide to patients. The domains are Mental Hospitals, Health units, Out-Patient Facilities, Day Treatment Centers, and Residential Facilities.

- Country
    - 112 Unique values holding names of countries around the world.
    - *String object*
- Year
    - Data for the year 2016 as the entire analysis was subjected to this specific year.
    - *Integer variable*
- Mental Hospitals
    - Number of mental hospitals for every 100K population within a country.
    - _Float variable
- Health Units
    - Number of mental health Units for every 100K population within a country.
    - *Float variable*
- Out-Patient Facilities
    - Number of out-patient facilities in the mental health sector for every 100K population within a country.
    - *Float variable*
- Day Treatment

- Number of mental health day treatment centers for every 100K population within a country.
- *Float variable*
- Residential Facilities
  - Number of community residential facilities for every 100K population within a country.
  - *Float variable*

Human Resources Dataset

This dataset holds the number of all Human Resource units available per every 100K population in their respective mental health sectors within each country across the world. The Human Resource units are categorized into 4 domains based on the type of service they provide to patients. The domains are Psychiatrists, Nurses, Social Workers, Psychologists.

- Country
  - 107 Unique values holding names of countries around the world.
  - *String object*
- Year
  - 2016 as a single year input as the entire analysis was subjected to this specific year.
  - *Integer variable*
- Psychiatrists
  - Number of Psychiatrists available for every 100K population within a country.
  - *Float variable*
- Nurses
  - Number of Nurses for every 100K population within a country.
  - *Float variable*
- Social Workers
  - Number of Social Workers for every 100K population within a country.
  - *Float variable*
- Psychologists
  - Number of Psychologists working in the mental health sector for every 100K population within a country.
  - *Float variable*

Overall, the data was substantially clean, just had a few missing values (<10%) in certain columns for facilities and resources datasets, which were taken into consideration during data cleaning and analysis. We should be mindful that the stigmatization of mental illness is still an important societal problem and as a consequence, number of resources for certain countries are still very limited. Despite these data quality issues, these datasets will give us as a snapshot of the different mental health faclities available to combat the high suicide rate in the world.

Project Aim and Objectives (5 marks)

A suicide attempt is a very tragic response to difficult situations and feelings and is a clear indication that someone is struggling and needs immediate help. The tragedy could be reduced or even prevented if adequate measures and support groups are made easily available. The general aim of our project is to provide a comprehensive analysis of the global suicide rate per 100K population through a certain length of time and classify how different parameters influence the global suicide rate trends. Finally, we intend to quantify different resources available by mental health care providers to tackle this universal challenge and design a model that can establish a correlation between them. We have implemented various visualisation tools at different stages to obtain a clear picture of the outcome of our analysis.

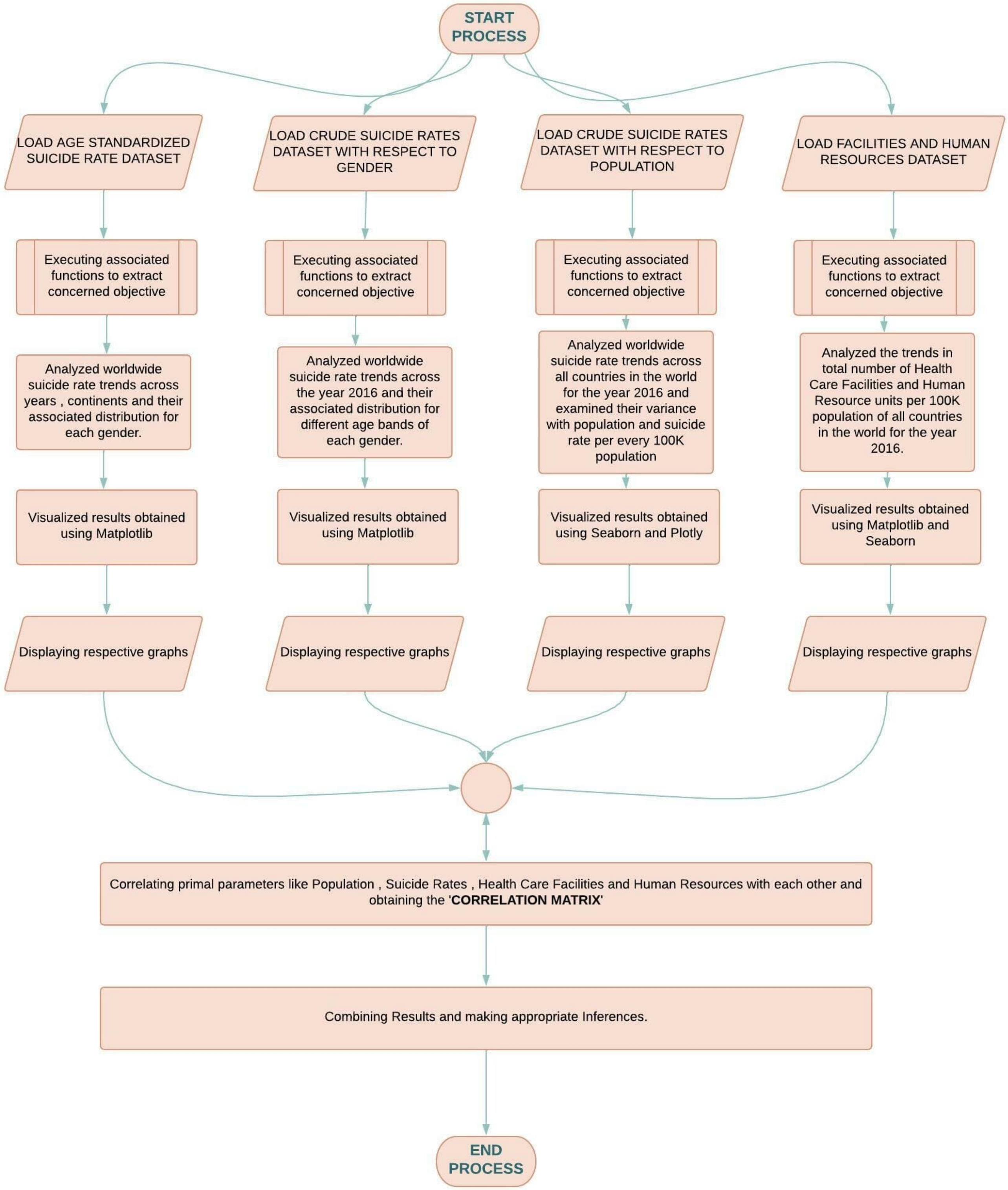
Specific Objective(s)

- **Objective 1:** *Year wise trends of suicide rates across 6 continents.*
- **Objective 2:** *Suicide rate in different genders and age groups.*
- **Objective 3:** *World map representation showing dispersal of the suicide rate.*
- **Objective 4:** *Distribution of human resource and mental health facilities across countries*

System Design (5 marks)

Architecture

The pipeline of our system is clearly illustrated in the flowchart below:



The flow of the process begins with the loading of corresponding datasets from our respective source. We have progressed with our coursework by defining various user-defined python functions necessary to target the task of each of our objectives. Within each function, we process the dataset according to the aim of its objective and analyze the trends/patterns in the results obtained. We complemented these results with appropriate visualizations to help us understand the variations in trends explicitly. Thereafter, We extended our findings by identifying how closely each of our primary parameters is related to each other and we accomplished this by using the packages from seaborn that helped us arrive at a correlation matrix. This whole system would help us to make suitable inferences that would prove beneficial for understanding suicide rates among people from various countries.

Processing Modules and Algorithms

- Data Cleaning using Pandas for any unusual outliers and duplicate or null values.
- Data Normalization using min- max scaling technique.
  - Formula: (X - column_minimum)/ (column_maximum-column_minimum), where X is the vallue to be normalised in the column
- Country to continent mapping done using pycountry_convert package.
- Installed and imported various interactive libraries like matplotlib, plotly, etc and used choropleth map, graph_objects features for better visualisation of our analysis.
- Obtained a correlation matrix between various columns using seaborn.

Program Code (15 marks)

Brief Explanation of following code cell

To carry out our analysis and export the notebook as a pdf, it was necessary to install few packages and import certain libraries. The following 2 sections of code includes all those details.

In [ ]:
```
# for country to continent mapping
!pip install pycountry_convert
!pip install pycountry
```

In [ ]:
```
!pip install -U notebook-as-pdf
!pip install nbconvert
!pip install pyppeteer
!pyppeteer-install
```

In [3]:
```
import plotly.io as pio
pio.renderers.default ="notebook"
```

In [4]:
```
import pandas as pd # to create dataframe from all datasets and carry out the analysis
import matplotlib.pyplot as plt # for visualisation purpose
import seaborn as sns # to plot heatmap for correlation matrix and various horizontal ba
import plotly.graph_objs as go # to plot choropleth map and pie  charts
from plotly.subplots import make_subplots # to make subplots
import pycountry # to get ISO 3166-1 alpha 3 codes from country name
from pycountry_convert import country_alpha2_to_continent_code, country_name_to_country_
```

Brief Explanation of following code cell

Here the first dataset is uploaded as a dataframe named ageSuicideDF and df.head() is called to show the columns of the dataframe.

In [5]:
```
#importing and displaying the dataset
ageSuicideDF = pd.read_csv('Age-standardized suicide rates.csv')
ageSuicideDF.head()
```

Out[5]:

	Country	Sex	2016	2015	2010	2000
0	Afghanistan	Both sexes	6.4	6.6	7.4	8.1
1	Afghanistan	Male	10.6	10.9	12.5	14.3
2	Afghanistan	Female	2.1	2.1	2.1	1.7
3	Albania	Both sexes	5.6	5.3	7.7	5.8
4	Albania	Male	7.0	6.7	9.5	8.2

Brief Explanation of following code cell

In the next 3 code cells, the dataframe is described followed by changing the "Sex" column to a categorical data type and checking the number of null and duplicate values in the dataframe.

In [6]:
```
ageSuicideDF.describe()
```

Out[6]:

	2016	2015	2010	2000
count	549.000000	549.000000	549.000000	549.000000
mean	9.792532	9.925683	10.544991	12.164117
std	7.469341	7.633241	8.620174	10.378318
min	0.000000	0.000000	0.000000	0.100000
25%	4.300000	4.400000	4.700000	5.100000
50%	7.900000	8.100000	8.400000	9.500000
75%	13.400000	13.300000	14.100000	16.000000
max	48.300000	51.500000	62.300000	85.800000

In [7]:
```
ageSuicideDF['Sex'] = pd.Categorical(ageSuicideDF.Sex)
ageSuicideDF.info() #check if any missing values are present
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 549 entries, 0 to 548
Data columns (total 6 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Country  549 non-null    object
 1   Sex      549 non-null    category
 2   2016     549 non-null    float64
 3   2015     549 non-null    float64
 4   2010     549 non-null    float64
 5   2000     549 non-null    float64
dtypes: category(1), float64(4), object(1)
memory usage: 22.2+ KB
```

In [8]:
```
#check for duplicate rows
duplicate_rows1 = ageSuicideDF.duplicated(keep = "first").sum()
if duplicate_rows1>0:
    print("Number of duplicate rows present:{}".format(duplicate_rows))
else:
    print("Duplicate rows not present")
```

```
Duplicate rows not present
```

Comment on previous cell output

The output clearly states that neither null nor duplicate values were present in the above dataframe.

Brief Explanation of following code cell

Here a new columm with values obtained after measuring overall suicide rate for each year is added to the dataframe.

In [9]:
```
ageSuicideDF["Overall_suicide_rate"] = ageSuicideDF.iloc[:, 2:-1].mean(axis = 1).round(1
ageSuicideDF.head()
```

Out[9]:

	Country	Sex	2016	2015	2010	2000	Overall_suicide_rate
0	Afghanistan	Both sexes	6.4	6.6	7.4	8.1	6.8
1	Afghanistan	Male	10.6	10.9	12.5	14.3	11.3
2	Afghanistan	Female	2.1	2.1	2.1	1.7	2.1
3	Albania	Both sexes	5.6	5.3	7.7	5.8	6.2
4	Albania	Male	7.0	6.7	9.5	8.2	7.7

Brief Explanation of following code cell

A function get_continent is defined that transforms country names to their respective continents.

In [10]:
```
def get_continent(col):
        try:
            cn_a2_code =  country_name_to_country_alpha2(col)
        except:
            cn_a2_code = 'Unknown'
        try:
            if cn_a2_code == "TL":
                cn_continent = "AS"
            else:
                cn_continent = country_alpha2_to_continent_code(cn_a2_code)
        except:
            cn_continent = 'Unknown'
        return (cn_continent)
```

Brief Explanation of following code cell

At first, 5 wrongly-formatted country names are reformatted as per ISO 3166-1 alpha-2 country list. Then, using the defined function, the continent code is obtained for each country, which is then converted to continent names.

In [11]:
```
#reformat country names to make them similar to ISO 3166-1 alpha-2 country list.
replace_values = {"Bolivia (Plurinational State of)": "Bolivia", "Iran (Islamic Republic
                      "Micronesia (Federated States of)": "Micronesia", "Republic of Kor
                      "Venezuela (Bolivarian Republic of)": "Venezuela", "CÃ´te d'Ivoire
ageSuicideDF["Country"] = ageSuicideDF["Country"].replace(replace_values)
country_list = list(ageSuicideDF["Country"])
continent_list = [get_continent(country) for country in country_list]

# addition of a continent column with continent names
ageSuicideDF['Continent'] = continent_list
continent_name = {"EU": "Europe", "AS": "Asia", "AF": "Africa", "NA": "North America",
                  "SA": "South America", "OC":"Oceania"}
ageSuicideDF["Continent"] = ageSuicideDF["Continent"].replace(continent_name)

ageSuicideDF.head()
```

Out[11]:

	Country	Sex	2016	2015	2010	2000	Overall_suicide_rate	Continent
0	Afghanistan	Both sexes	6.4	6.6	7.4	8.1	6.8	Asia
1	Afghanistan	Male	10.6	10.9	12.5	14.3	11.3	Asia
2	Afghanistan	Female	2.1	2.1	2.1	1.7	2.1	Asia
3	Albania	Both sexes	5.6	5.3	7.7	5.8	6.2	Europe
4	Albania	Male	7.0	6.7	9.5	8.2	7.7	Europe

Comment on previous cell output

The dataframe has 2 columns added : Overall_suicide_rate and Continent based on the above analysis.

Brief Explanation of following code cell

Here the overall suicide rate of each year is measured to see how the rate has changed over the 4 years.

In [12]:
```
#total suicide rate per year
yearlyDF = pd.DataFrame()
columnSum = ageSuicideDF[["2016", "2015", "2010", "2000"]].mean().round(1)
yearlyDF['Total Suicide Rate'] = columnSum
yearlyDF.head()
```

Out[12]:

	Total Suicide Rate
2016	9.8
2015	9.9
2010	10.5
2000	12.2

Comment on previous cell output

Here the output shows the total suicide rate per 100K for each year in ascending order.

Brief Explanation of following code cell

A function, lineGraphYearlySuicideRate() is defined that has all the codes necessary to generate a line plot for the above shown analysis.

In [13]:
```
def lineGraphYearlySuicideRate():
    plt.figure(figsize=(10,7))
    plt.plot(yearlyDF[::-1], marker ='*', color='red')

    plt.title("Suicide Rate yearly", fontdict = {'fontsize' : 20, "color" : "Blue"})
    plt.xlabel("Years", fontdict = {'fontsize' : 15})
    plt.ylabel("Suicide Rate", fontdict = {'fontsize' : 15})
    plt.show()
```

Brief Explanation of following code cell

In the next code, the suicide rate has been grouped by different genders to exhibit their distribution for all 4 years.

In [14]:
```
genderDF = pd.DataFrame()
genderGroup = ageSuicideDF.groupby(['Sex']).mean().reset_index().round(1)
genderGroup
```

Out[14]:

	Sex	2016	2015	2010	2000	Overall_suicide_rate
0	Both sexes	9.7	9.8	10.4	12.0	10.0
1	Female	5.1	5.2	5.5	6.4	5.3
2	Male	14.6	14.8	15.7	18.1	15.0

Comment on previous cell output

The output summarizes the changes in suicide rate for all 3 genders across the 4 years.

Brief Explanation of following code cell

To visualize the trend in suicide rate, by gender, a function barGraphGenderSuicideRate() is defined which generates a bar plot for all the years.

In [15]:
```
def barGraphGenderSuicideRate(ageSuicideDF):
    plt.rcParams["figure.figsize"] = (10,6)
    testDF = pd.DataFrame()
    genderGroup = ageSuicideDF.groupby(['Sex'])
    for gender, genderInfo  in genderGroup:
        genderDF = genderGroup.get_group(gender)
        columnsSum = genderDF[["2016", "2015", "2010", "2000"]].mean()
        testDF[gender] = columnsSum
    genderDF = testDF
    genderDF.plot(kind="bar")
    plt.title("Gender wise Suicide Rate yearly", fontdict = {'fontsize' : 20, "color" :
    plt.xlabel("Years", fontdict = {'fontsize' : 15})
    plt.ylabel("Suicide Rate", fontdict = {'fontsize' : 15})
```

Brief Explanation of following code cell

In the following code cell, the dataframe is filtered by year and continent columns, then grouped by continent and finally, the average suicide rate of all the continents is measured for all 4 years.

In [16]:
```
# Year 2000
Filtered_DF1 = ageSuicideDF[['2000','Continent']]
A_2000 = Filtered_DF1.groupby(by='Continent')['2000'].sum()
D_2000 = Filtered_DF1.groupby(by='Continent')['2000'].sum().sum()
percentage_2000 = (A_2000 / D_2000 ) * 100
K1 =percentage_2000.sort_values(ascending = False)

#Year 2010
Filtered_DF2 = ageSuicideDF[['2010','Continent']]
A_2010 = Filtered_DF2.groupby(by='Continent')['2010'].sum()
D_2010 = Filtered_DF2.groupby(by='Continent')['2010'].sum().sum()
percentage_2010 = ( A_2010 / D_2010) * 100
K2 =percentage_2010.sort_values(ascending = False)


#Year 2015
Filtered_DF3 = ageSuicideDF[['2015','Continent']]
A_2015 = Filtered_DF3.groupby(by='Continent')['2015'].sum()
D_2015 = Filtered_DF3.groupby(by='Continent')['2015'].sum().sum()
percentage_2015 = ( A_2015 / D_2015) * 100
K3 =percentage_2015.sort_values(ascending = False)

# Year 2016
Filtered_DF4 = ageSuicideDF[['2016','Continent']]
A_2016 = Filtered_DF4.groupby(by='Continent')['2016'].sum()
D_2016 = Filtered_DF4.groupby(by='Continent')['2016'].sum().sum()
percentage_2016 = ( A_2016 / D_2016) * 100
K4 =percentage_2016.sort_values(ascending = False)

K1
```

Out[16]:
```
Continent
```

```
Africa             31.344245
Europe             29.583265
Asia               19.881403
North America       7.548554
South America       6.993007
Oceania             4.649526
Name: 2000, dtype: float64
```

Comment on previous cell output

The output reveals the suicide rate distribution in descending order across all continents for the year 2000. Only one year is displayed here to demonstrate the analysis.

Brief Explanation of following code cell

A function lineGraphContinentalSuicideRate() is defined that includes codes to plot an overlap of line graphs for all 4 years across all 6 continents.

In [17]:
```
def lineGraphContinentalSuicideRate():
    plt.figure(figsize=(18,6))
    plt.plot(K1, marker='*', label='Year 2000', color="c")
    plt.plot(K2, marker='*', label='Year 2010', color="m")
    plt.plot(K3, marker='*', label='Year 2015', color="g")
    plt.plot(K4, marker='*', label='Year 2016', color="r")
    plt.legend()
    plt.xlabel('Continents', fontdict = {'fontsize' : 15})
    plt.ylabel('Suicide Rate %', fontdict = {'fontsize' : 15})
    plt.title('Year wise trends in Suicide Rate',fontdict = {'fontsize' : 20, "color" :
    plt.show()
```

Brief Explanation of following code cell

After the analysis of first dataset, second dataset is uploaded as a dataframe as genderSuicideDF. Then df.head() is called to show the columns of the dataframe.

In [18]:
```
#importing and displaying the dataset
genderSuicideDF = pd.read_csv("Crude suicide rates.csv")
genderSuicideDF.head()
```

Out[18]:

	Country	Sex	80_above	70to79	60to69	50to59	40to49	30to39	20to29	10to19
0	Afghanistan	Both sexes	42.0	11.0	5.5	5.6	6.6	9.2	10.2	3.1
1	Afghanistan	Male	70.4	20.9	9.8	9.3	10.5	15.1	16.3	4.8
2	Afghanistan	Female	20.1	2.3	1.4	1.6	2.3	2.7	3.5	1.2
3	Albania	Both sexes	16.3	8.3	6.0	7.8	9.1	6.1	6.5	5.0
4	Albania	Male	23.2	11.9	8.1	11.4	13.5	8.8	6.3	3.1

Comment on previous cell output

The datframe shows suicide rate for different age bands in the year 2016 for 3 different genders.

Brief Explanation of following code cell

Before initiating analysis, initially the dataframe is described followed by changing the "Sex" column to a categorical data type and checking the number of null and duplicate values in the dataframe in the next 3 code cells.

In [19]:
```
genderSuicideDF.describe()
```

Out[19]:

	80_above	70to79	60to69	50to59	40to49	30to39	20to29	10to19
count	549.000000	549.000000	549.000000	549.000000	549.000000	549.000000	549.000000	549.000000
mean	42.585428	25.936794	17.439162	14.743352	12.189435	10.895446	10.423315	4.075046
std	43.477900	24.795457	14.315504	12.790041	11.121364	10.148280	9.051273	3.393507
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	13.600000	8.400000	6.600000	5.600000	4.800000	4.300000	4.300000	1.700000
50%	26.600000	17.100000	12.900000	11.200000	9.000000	8.000000	7.900000	3.100000
75%	55.500000	35.800000	24.000000	20.000000	16.400000	13.900000	13.500000	5.300000
max	285.000000	133.700000	78.500000	85.400000	86.500000	88.400000	57.400000	24.200000

In [20]:
```
genderSuicideDF['Sex'] = pd.Categorical(genderSuicideDF.Sex)
genderSuicideDF.info() #check if any missing values are present
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 549 entries, 0 to 548
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Country    549 non-null    object
 1   Sex        549 non-null    category
 2    80_above  549 non-null    float64
 3    70to79    549 non-null    float64
 4    60to69    549 non-null    float64
 5    50to59    549 non-null    float64
 6    40to49    549 non-null    float64
 7    30to39    549 non-null    float64
 8    20to29    549 non-null    float64
 9    10to19    549 non-null    float64
dtypes: category(1), float64(8), object(1)
memory usage: 39.4+ KB
```

In [21]:
```
#check for duplicate rows
duplicate_rows1 = genderSuicideDF.duplicated(keep = "first").sum()
if duplicate_rows1>0:
    print("Number of duplicate rows present:{}".format(duplicate_rows))
else:
    print("Duplicate rows not present")
```

```
Duplicate rows not present
```

Comment on previous cell output

The output of above 2 code cells confirms that neither null nor duplicate values were present in the above dataframe.

Brief Explanation of following code cell

Here a new columm with values obtained after measuring overall suicide rate for each year is added to the dataframe.

In [22]:
```
genderSuicideDF["Overall_suicide_rate"] = genderSuicideDF.iloc[:, 2:-1].mean(axis = 1).r
```

```
genderSuicideDF.head()
```

Out[22]:

	Country	Sex	80_above	70to79	60to69	50to59	40to49	30to39	20to29	10to19	Overall_suicide_rate
0	Afghanistan	Both sexes	42.0	11.0	5.5	5.6	6.6	9.2	10.2	3.1	12.9
1	Afghanistan	Male	70.4	20.9	9.8	9.3	10.5	15.1	16.3	4.8	21.8
2	Afghanistan	Female	20.1	2.3	1.4	1.6	2.3	2.7	3.5	1.2	4.8
3	Albania	Both sexes	16.3	8.3	6.0	7.8	9.1	6.1	6.5	5.0	8.6
4	Albania	Male	23.2	11.9	8.1	11.4	13.5	8.8	6.3	3.1	11.9

Comment on previous cell output

The "Overall_suicide_rate" column in this genderSuicideDF data measures the average suicide rate per 100K for each age band and each gender for individual country.

Brief Explanation of following code cell

In the next 2 cells, the total suicide rate for each gender is calculated by grouping the "Sex" column variable over the "Overall_suicide_rate" column. After this, the percentage of all 3 genders was evaluated for a better comparison.

In [23]:
```
Filtered_DF5 = genderSuicideDF.groupby(by= 'Sex')['Overall_suicide_rate'].mean().round(1
Filtered_DF5.head()
```

Out[23]:
```
Sex
 Both sexes    18.5
 Female        10.3
 Male          28.7
Name: Overall_suicide_rate, dtype: float64
```

In [24]:
```
#calculating % of suicide rate for each gender
denominator = Filtered_DF5[0] + Filtered_DF5[1]+Filtered_DF5[2]
Percentage_male= (Filtered_DF5[2] /( denominator) )* 100
Percentage_Female= (Filtered_DF5[1] /(denominator) )* 100
Percentage_Both_Sexes = (Filtered_DF5[0] /( denominator) )* 100
print( ' Male = ' ,Percentage_male , ' Female = ',Percentage_Female  , ' Both sexes = ',
```

```
 Male =  49.91304347826087  Female =  17.91304347826087  Both sexes =  32.17391304347826
```

Brief Explanation of following code cell

A function, donutChartGender() is defined that has all the codes necessary to generate a donut chart for the above shown analysis using matplotlib.

In [25]:
```
def donutChartGender():
    Gender = ['Male' , 'Female', 'Both Sexes' ]
    Suicide_rate = [Percentage_male ,Percentage_Female, Percentage_Both_Sexes ]
    colors = ['#FFFF00', '#ADFF2F', '#FF0000']
    # explosion
    explode = (0.05, 0.05, 0.05)
    plt.figure(figsize=(18,6))
    # Pie Chart
    plt.pie(Suicide_rate, colors=colors, labels=Gender,
            autopct='%1.1f%%', pctdistance=0.85,
            explode=explode, textprops={'fontsize': 14})
```

```
    # draw circle
    centre_circle = plt.Circle((0, 0), 0.70, fc='White')
    fig = plt.gcf()
    # Adding Circle in Pie chart
    fig.gca().add_artist(centre_circle)
    # Adding Title of chart
    plt.title('Gender v/s Suicide Rate Worldwide', fontdict = {'fontsize' : 20, "color"
    return(plt.show())
```

Brief Explanation of following code cell

In the next code, the suicide rate has been grouped by different age groups for each gender to find out how different ages impact suicide rates for each gender. Following each code cell, there is another cell that contains the necessary codes to plot a line graph based on the results of the analysis code.

- _Male: age groups v/s suicide rate*analysis code cell*

In [26]:
```
pd.options.mode.chained_assignment = None
malelist = genderSuicideDF["Sex"].isin([" Male"])
# extracting data with gender = male only
genderSuicideDF_male=genderSuicideDF[malelist]

# 80_above normalisation
maximim = max(genderSuicideDF_male[' 80_above'])
minimum = min(genderSuicideDF_male[' 80_above'])
pd.set_option("display.precision", 4)
genderSuicideDF_male['normalised 80_above'] =(( genderSuicideDF_male[[' 80_above']]- min
Final_80_Above_male=genderSuicideDF_male['normalised 80_above'].sum().round(1)

#70to79
maximim = max(genderSuicideDF_male[' 70to79'])
minimum = min(genderSuicideDF_male[' 70to79'])
pd.set_option("display.precision", 4)
genderSuicideDF_male['normalised 70to79'] =(( genderSuicideDF_male[[' 70to79']]- minimum
Final_70to79_male=genderSuicideDF_male['normalised 70to79'].sum().round(1)

#60to69
maximim = max(genderSuicideDF_male[' 60to69 '])
minimum = min(genderSuicideDF_male[' 60to69 '])
pd.set_option("display.precision", 4)
genderSuicideDF_male['normalised 60to69'] =(( genderSuicideDF_male[[' 60to69 ']]- minimu
Final_60to69_male=genderSuicideDF_male['normalised 60to69'].sum().round(1)

#50to59
maximim = max(genderSuicideDF_male[' 50to59 '])
minimum = min(genderSuicideDF_male[' 50to59 '])
pd.set_option("display.precision", 4)
genderSuicideDF_male['normalised 50to59'] =(( genderSuicideDF_male[[' 50to59 ']]- minimu
Final_50to59_male=genderSuicideDF_male['normalised 50to59'].sum().round(1)

#40to49
maximim = max(genderSuicideDF_male[' 40to49'])
minimum = min(genderSuicideDF_male[' 40to49'])
pd.set_option("display.precision", 4)
genderSuicideDF_male['normalised 40to49'] =(( genderSuicideDF_male[[' 40to49']]- minimum
Final_40to49_male=genderSuicideDF_male['normalised 40to49'].sum().round(1)

#30to39
maximim = max(genderSuicideDF_male[' 30to39'])
minimum = min(genderSuicideDF_male[' 30to39'])
pd.set_option("display.precision", 4)
```

```
genderSuicideDF_male['normalised 30to39'] =(( genderSuicideDF_male[[' 30to39']]- minimum
Final_30to39_male=genderSuicideDF_male['normalised 30to39'].sum().round(1)

#20to29
maximim = max(genderSuicideDF_male[' 20to29'])
minimum = min(genderSuicideDF_male[' 20to29'])
pd.set_option("display.precision", 4)
genderSuicideDF_male['normalised 20to29'] =(( genderSuicideDF_male[[' 20to29']]- minimum
Final_20to29_male=genderSuicideDF_male['normalised 20to29'].sum().round(1)

#10to19
maximim = max(genderSuicideDF_male[' 10to19'])
minimum = min(genderSuicideDF_male[' 10to19'])
pd.set_option("display.precision", 4)
genderSuicideDF_male['normalised 10to19'] =(( genderSuicideDF_male[[' 10to19']]- minimum
Final_10to19_male=genderSuicideDF_male['normalised 10to19'].sum().round(1)
print("10to19_male: ", Final_10to19_male, ", 20to29_male: ", Final_20to29_male, ", 30to3
print("40to49_male: ", Final_40to49_male, ", 50to59_male: ", Final_50to59_male, ", 60to6
print("70to79_male: ", Final_70to79_male, ", 80_above_male: ", Final_80_Above_male)
```

```
10to19_male:  40.0 , 20to29_male:  50.1 , 30to39_male:  34.8
40to49_male:  39.7 , 50to59_male:  47.5 , 60to69_male:  58.9
70to79_male:  55.7 , 80_above_male:  41.4
```

- _Male: age groups v/s suicide rate*line plot code cell*

In [27]:

```
def lineGraphMale():
    x_axis = ['80_above', '70to79','60to69', '50to59','40to49','30to39','20to29','10to19
    y_axis = [Final_80_Above_male,Final_70to79_male,Final_60to69_male, Final_50to59_male
              Final_30to39_male,Final_20to29_male, Final_10to19_male]
    plt.figure(figsize=(10,3))
    plt.plot(x_axis, y_axis, marker='.')
    plt.title('Age-wise global suicide rate for male', fontdict = {'fontsize' : 20, "col
    plt.xlabel('age group', fontdict = {'fontsize' : 15})
    plt.ylabel('suicide rate', fontdict = {'fontsize' : 15})
    return(plt.show())
```

- _Female: age groups v/s suicide rate*analysis code cell*

In [28]:

```
femalelist = genderSuicideDF["Sex"].isin([" Female"])
# extracting data with gender = male only
genderSuicideDF_female=genderSuicideDF[femalelist]

# 80_above normalisation
maximim = max(genderSuicideDF_female[' 80_above'])
minimum = min(genderSuicideDF_female[' 80_above'])
pd.set_option("display.precision", 4)
genderSuicideDF_female['normalised 80_above'] =(( genderSuicideDF_female[[' 80_above']]-
Final_80_Above_female=genderSuicideDF_female['normalised 80_above'].sum().round(1)

#70to79
maximim = max(genderSuicideDF_female[' 70to79'])
minimum = min(genderSuicideDF_female[' 70to79'])
pd.set_option("display.precision", 4)
genderSuicideDF_female['normalised 70to79'] =(( genderSuicideDF_female[[' 70to79']]- min
Final_70to79_female=genderSuicideDF_female['normalised 70to79'].sum().round(1)

#60to69
maximim = max(genderSuicideDF_female[' 60to69 '])
minimum = min(genderSuicideDF_female[' 60to69 '])
pd.set_option("display.precision", 4)
genderSuicideDF_female['normalised 60to69'] =(( genderSuicideDF_female[[' 60to69 ']]- mi
Final_60to69_female=genderSuicideDF_female['normalised 60to69'].sum().round(1)
```

```
#50to59
maximim = max(genderSuicideDF_female[' 50to59 '])
minimum = min(genderSuicideDF_female[' 50to59 '])
pd.set_option("display.precision", 4)
genderSuicideDF_female['normalised 50to59'] =(( genderSuicideDF_female[[' 50to59 ']]- mi
Final_50to59_female=genderSuicideDF_female['normalised 50to59'].sum().round(1)

#40to49
maximim = max(genderSuicideDF_female[' 40to49'])
minimum = min(genderSuicideDF_female[' 40to49'])
pd.set_option("display.precision", 4)
genderSuicideDF_female['normalised 40to49'] =(( genderSuicideDF_female[[' 40to49']]- min
Final_40to49_female=genderSuicideDF_female['normalised 40to49'].sum().round(1)

#30to39
maximim = max(genderSuicideDF_female[' 30to39'])
minimum = min(genderSuicideDF_female[' 30to39'])
pd.set_option("display.precision", 4)
genderSuicideDF_female['normalised 30to39'] =(( genderSuicideDF_female[[' 30to39']]- min
Final_30to39_female=genderSuicideDF_female['normalised 30to39'].sum().round(1)

#20to29
maximim = max(genderSuicideDF_female[' 20to29'])
minimum = min(genderSuicideDF_female[' 20to29'])
pd.set_option("display.precision", 4)
genderSuicideDF_female['normalised 20to29'] =(( genderSuicideDF_female[[' 20to29']]- min
Final_20to29_female=genderSuicideDF_female['normalised 20to29'].sum().round(1)

#10to19
maximim = max(genderSuicideDF_female[' 10to19'])
minimum = min(genderSuicideDF_female[' 10to19'])
pd.set_option("display.precision", 4)
genderSuicideDF_female['normalised 10to19'] =(( genderSuicideDF_female[[' 10to19']]- min
Final_10to19_female=genderSuicideDF_female['normalised 10to19'].sum().round(1)
print("10to19_female: ",Final_10to19_female,", 20to29_female: ",Final_20to29_female, ",3
print("40to49_female: ",Final_40to49_female,", 50to59_female: ",Final_50to59_female,", 6
print("70to79_female: ", Final_70to79_female, ", 80_above_female: ", Final_80_Above_fema
```

```
10to19_female:  28.3 , 20to29_female:  27.9 ,30to39_female:  33.8
40to49_female:  25.3 , 50to59_female:  24.1 , 60to69_female:  26.1
70to79_female:  21.1 , 80_above_female:  20.6
```

- _Female: age groups v/s suicide rate*line plot code cell*

In [29]:
```
def lineGraphFemale():
    x_axis = ['80_above', '70to79','60to69', '50to59','40to49','30to39','20to29','10to19
    y_axis = [Final_80_Above_female,Final_70to79_female,Final_60to69_female, Final_50to5
              Final_30to39_female,Final_20to29_female, Final_10to19_female]
    plt.figure(figsize=(10,3))
    plt.plot(x_axis, y_axis, marker='.', color='m')
    plt.title('Age-wise global suicide rate for female', fontdict = {'fontsize' : 20, "c
    plt.xlabel('age group', fontdict = {'fontsize' : 15})
    plt.ylabel('suicide rate', fontdict = {'fontsize' : 15})
    return(plt.show())
```

- _Both Sexes: age groups v/s suicide rate*analysis code cell*

In [30]:
```
bothsexes_list = genderSuicideDF["Sex"].isin([" Both sexes"])
# extracting data with gender = male only
genderSuicideDF_bs=genderSuicideDF[bothsexes_list]

# 80_above normalisation
```

```
maximim = max(genderSuicideDF_bs[' 80_above'])
minimum = min(genderSuicideDF_bs[' 80_above'])
pd.set_option("display.precision", 4)
genderSuicideDF_bs['normalised 80_above'] =(( genderSuicideDF_bs[[' 80_above']]- minimum
Final_80_Above_bs=genderSuicideDF_bs['normalised 80_above'].sum().round(1)

#70to79
maximim = max(genderSuicideDF_bs[' 70to79'])
minimum = min(genderSuicideDF_bs[' 70to79'])
pd.set_option("display.precision", 4)
genderSuicideDF_bs['normalised 70to79'] =(( genderSuicideDF_bs[[' 70to79']]- minimum) /
Final_70to79_bs=genderSuicideDF_bs['normalised 70to79'].sum().round(1)

#60to69
maximim = max(genderSuicideDF_bs[' 60to69 '])
minimum = min(genderSuicideDF_bs[' 60to69 '])
pd.set_option("display.precision", 4)
genderSuicideDF_bs['normalised 60to69'] =(( genderSuicideDF_bs[[' 60to69 ']]- minimum) /
Final_60to69_bs=genderSuicideDF_bs['normalised 60to69'].sum().round(1)

#50to59
maximim = max(genderSuicideDF_bs[' 50to59 '])
minimum = min(genderSuicideDF_bs[' 50to59 '])
pd.set_option("display.precision", 4)
genderSuicideDF_bs['normalised 50to59'] =(( genderSuicideDF_bs[[' 50to59 ']]- minimum) /
Final_50to59_bs=genderSuicideDF_bs['normalised 50to59'].sum().round(1)

#40to49
maximim = max(genderSuicideDF_bs[' 40to49'])
minimum = min(genderSuicideDF_bs[' 40to49'])
pd.set_option("display.precision", 4)
genderSuicideDF_bs['normalised 40to49'] =(( genderSuicideDF_bs[[' 40to49']]- minimum) /
Final_40to49_bs=genderSuicideDF_bs['normalised 40to49'].sum().round(1)

#30to39
maximim = max(genderSuicideDF_bs[' 30to39'])
minimum = min(genderSuicideDF_bs[' 30to39'])
pd.set_option("display.precision", 4)
genderSuicideDF_bs['normalised 30to39'] =(( genderSuicideDF_bs[[' 30to39']]- minimum) /
Final_30to39_bs=genderSuicideDF_bs['normalised 30to39'].sum().round(1)

#20to29
maximim = max(genderSuicideDF_bs[' 20to29'])
minimum = min(genderSuicideDF_bs[' 20to29'])
pd.set_option("display.precision", 4)
genderSuicideDF_bs['normalised 20to29'] =(( genderSuicideDF_bs[[' 20to29']]- minimum) /
Final_20to29_bs=genderSuicideDF_bs['normalised 20to29'].sum().round(1)

#10to19
maximim = max(genderSuicideDF_bs[' 10to19'])
minimum = min(genderSuicideDF_bs[' 10to19'])
pd.set_option("display.precision", 4)
genderSuicideDF_bs['normalised 10to19'] =(( genderSuicideDF_bs[[' 10to19']]- minimum) /
Final_10to19_bs=genderSuicideDF_bs['normalised 10to19'].sum().round(1)
print("10to19_bs: ",Final_10to19_bs,", 20to29_bs: ",Final_20to29_bs, ",30to39_bs: ",Fina
print("40to49_bs: ",Final_40to49_bs,", 50to59_bs: ",Final_50to59_bs,", 60to69_bs: ",Fina
print("70to79_bs: ", Final_70to79_bs, ", 80_above_bs: ", Final_80_Above_bs)
```

```
10to19_bs:  40.9 , 20to29_bs:  50.6 ,30to39_bs:  40.2
40to49_bs:  40.6 , 50to59_bs:  55.0 , 60to69_bs:  56.4
70to79_bs:  46.0 , 80_above_bs:  35.2
```

- _Both sexes: age groups v/s suicide rate*line plot code cell*

In [31]:
```
def lineGraphBothSexes():
    x_axis = ['80_above', '70to79','60to69', '50to59','40to49','30to39','20to29','10to19
    y_axis = [Final_80_Above_bs,Final_70to79_bs,Final_60to69_bs, Final_50to59_bs,
              Final_40to49_bs, Final_30to39_bs,Final_20to29_bs, Final_10to19_bs]
    plt.figure(figsize=(10,3))
    plt.plot(x_axis, y_axis, marker='.', color='green')
    plt.title('Age-wise global suicide rate for both sexes', fontdict = {'fontsize' : 20
    plt.xlabel('age group', fontdict = {'fontsize' : 15})
    plt.ylabel('suicide rate', fontdict = {'fontsize' : 15})
    return(plt.show())
```

Brief Explanation of following code cell

To find countries with the highest suicide rates, a third dataset with added population column to the second dataset is uploaded and transformed to a new dataframe, and df.head() is called to show the columns of the dataframe.

In [32]:
```
suicidePopulationDF = pd.read_csv("Suicide rate and Population together.csv")
suicidePopulationDF.head()
```

Out[32]:

	Country	Sex	80_above	70to79	60to69	50to59	40to49	30to39	20to29	10to19	Population 2016 in millions
0	Afghanistan	Both sexes	42.0	11.0	5.5	5.6	6.6	9.2	10.2	3.1	33.4
1	Afghanistan	Male	70.4	20.9	9.8	9.3	10.5	15.1	16.3	4.8	33.4
2	Afghanistan	Female	20.1	2.3	1.4	1.6	2.3	2.7	3.5	1.2	33.4
3	Albania	Both sexes	16.3	8.3	6.0	7.8	9.1	6.1	6.5	5.0	2.9
4	Albania	Male	23.2	11.9	8.1	11.4	13.5	8.8	6.3	3.1	2.9

Brief Explanation of following code cell

Before initiating analysis, like the first 2 datasets, firstly the dataframe is described in the next 3 code cells, followed by changing the "Sex" column to a categorical data type and checking the number of null and duplicate values in the dataframe.

In [33]:
```
suicidePopulationDF.describe()
```

Out[33]:

	80_above	70to79	60to69	50to59	40to49	30to39	20to29	10to19	Population 2016 in millions
count	549.0000	549.0000	549.0000	549.0000	549.0000	549.0000	549.0000	549.0000	549.0000
mean	42.5854	25.9368	17.4392	14.7434	12.1894	10.8954	10.4233	4.0750	40.2677
std	43.4779	24.7955	14.3155	12.7900	11.1214	10.1483	9.0513	3.3935	145.6004
min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0900
25%	13.6000	8.4000	6.6000	5.6000	4.8000	4.3000	4.3000	1.7000	2.5000
50%	26.6000	17.1000	12.9000	11.2000	9.0000	8.0000	7.9000	3.1000	9.3000
75%	55.5000	35.8000	24.0000	20.0000	16.4000	13.9000	13.5000	5.3000	28.4000
max	285.0000	133.7000	78.5000	85.4000	86.5000	88.4000	57.4000	24.2000	1378.0000

In [34]:
```
suicidePopulationDF['Sex'] = pd.Categorical(suicidePopulationDF.Sex)
suicidePopulationDF.info() #check if any missing values are present
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 549 entries, 0 to 548
Data columns (total 11 columns):
 #   Column                      Non-Null Count  Dtype   
---  ------                      --------------  -----   
 0   Country                     549 non-null    object  
 1   Sex                         549 non-null    category
 2    80_above                   549 non-null    float64 
 3    70to79                     549 non-null    float64 
 4    60to69                     549 non-null    float64 
 5    50to59                     549 non-null    float64 
 6    40to49                     549 non-null    float64 
 7    30to39                     549 non-null    float64 
 8    20to29                     549 non-null    float64 
 9    10to19                     549 non-null    float64 
 10  Population 2016 in millions  549 non-null    float64 
dtypes: category(1), float64(9), object(1)
memory usage: 43.7+ KB
```

In [35]:
```
duplicate_rows = suicidePopulationDF.duplicated(keep = "first").sum()
if duplicate_rows>0:
    print("Number of duplicate rows present:{}".format(duplicate_rows))
else:
    print("Duplicate rows not present")
```

```
Duplicate rows not present
```

Comment on previous cell output

The output of above code cells confirms that dataframe do not have any missing and duplicate values.

Brief Explanation of following code cell

In the code below, the dataframe is grouped by the "Country" column to incorporate the suicide rate contribution of all genders for each country.

In [36]:
```
#combine suicide rates for all 3 genders for each country
suicideData_cp = suicidePopulationDF.copy()
countryGroup = suicideData_cp.groupby('Country').mean().reset_index()
countryGroup.head()
```

Out[36]:

	Country	80_above	70to79	60to69	50to59	40to49	30to39	20to29	10to19	Population 2016 in millions
0	Afghanistan	44.1667	11.4000	5.5667	5.5000	6.4667	9.0000	10.0000	3.0333	33.40
1	Albania	16.8000	8.3667	6.0000	7.8667	9.2000	6.1000	6.4667	5.0333	2.90
2	Algeria	9.5000	5.6667	4.2000	4.1000	4.7000	5.2667	4.1667	1.3000	40.80
3	Angola	70.6667	44.0000	24.3333	15.0667	7.0667	5.4333	6.6333	2.6000	25.80
4	Antigua and Barbuda	0.0000	0.0000	7.9667	0.0000	0.0000	0.0000	0.0000	0.0000	0.09

Brief Explanation of following code cell

Here, a new column, "Suicide Rate per 100K" is created that consists of the average suicide rates of all countries including all age groups.

In [37]:
```
countryGroup["Suicide Rate per 100K"] = countryGroup.iloc[:, 1:-2].mean(axis=1).round(1)
sortedDF = countryGroup.sort_values(by="Suicide Rate per 100K", ascending=False)
sortedDF.head(10)
```

Out[37]:

	Country	80_above	70to79	60to69	50to59	40to49	30to39	20to29	10to19	Population 2016 in millions	Suicide Rate per 100K
93	Lesotho	151.9667	92.0667	52.8000	46.3667	35.4000	28.6667	23.2667	9.7000	2.2	61.5
182	Zimbabwe	214.3667	84.1333	43.5000	30.8000	19.5667	14.0000	11.4000	4.5667	16.0	59.7
169	Uganda	155.7000	98.2333	47.3000	29.6667	17.9000	14.5000	14.3000	5.3333	36.6	53.9
43	CÃ´te d'Ivoire	128.4333	86.0333	50.7000	32.9667	23.4000	21.1000	20.1000	8.0667	23.9	51.8
29	Cameroon	118.0333	79.1000	46.6000	29.3000	18.8000	17.6000	16.5333	6.3000	24.4	46.6
119	Nigeria	127.2000	86.5000	48.6333	26.0333	14.6000	12.1000	11.4000	4.0000	186.5	46.6
163	Togo	129.2000	78.4000	43.6667	25.0000	15.6667	12.9000	10.6667	4.0333	7.5	45.1
52	Equatorial Guinea	99.6333	68.6667	41.2000	29.7000	22.2333	23.9667	24.0000	7.0667	0.9	44.2
96	Lithuania	63.5333	44.3333	44.0667	47.9000	43.0667	36.2333	26.6333	8.0000	2.9	43.7
131	Republic of Korea	96.7000	67.4333	34.6333	35.6667	30.4000	25.8333	14.0667	4.4000	50.8	43.5

Comment on previous cell output

The output is displaying 10 countries with highest suicide rate per 100K in the world.

Brief Explanation of following code cell

A function alpha3code is defined which converts country names to ISO 3166-1 alpha-3 codes using pycountry package. The alpha-3 codes are added to the "Country Code" column of the sortedDF.

In [38]:
```
def alpha3code(column):
    CODE=[]
    for country in column:
        try:
            code=pycountry.countries.get(name=country).alpha_3
            CODE.append(code)
        except:
            CODE.append('None')
    return CODE
```

In [39]:
```
sortedDF["Country"] = sortedDF["Country"].replace(replace_values)
countryList = list(sortedDF["Country"])
sortedDF["Country Code"] = alpha3code(sortedDF.Country)
sortedDF.head()
```

Out[39]:

	Country	80_above	70to79	60to69	50to59	40to49	30to39	20to29	10to19	Population 2016 in millions	Suicide Rate per 100K	Cou
93	Lesotho	151.9667	92.0667	52.8	46.3667	35.4000	28.6667	23.2667	9.7000	2.2	61.5	
182	Zimbabwe	214.3667	84.1333	43.5	30.8000	19.5667	14.0000	11.4000	4.5667	16.0	59.7	

169	Uganda	155.7000	98.2333	47.3	29.6667	17.9000	14.5000	14.3000	5.3333	36.6	53.9	
43	Côte d'Ivoire	128.4333	86.0333	50.7	32.9667	23.4000	21.1000	20.1000	8.0667	23.9	51.8	
29	Cameroon	118.0333	79.1000	46.6	29.3000	18.8000	17.6000	16.5333	6.3000	24.4	46.6	

Brief Explanation of following code cell

The countries are converted to their respective continents using the previously defined get_continent function and the output is added to the "Continent" column of the sortedDF.

In [40]:
```
continentList = [get_continent(country) for country in countryList]
sortedDF['Continent'] = continentList
#continent_name = {"EU": "Europe", "AS": "Asia", "AF": "Africa", "NA":
#                  "North America", "SA": "South America", "OC":"Oceania"}
sortedDF["Continent"] = sortedDF["Continent"].replace(continent_name)
sortedDF.head()
```

Out[40]:

	Country	80_above	70to79	60to69	50to59	40to49	30to39	20to29	10to19	Population 2016 in millions	Suicide Rate per 100K	Cou
93	Lesotho	151.9667	92.0667	52.8	46.3667	35.4000	28.6667	23.2667	9.7000	2.2	61.5	
182	Zimbabwe	214.3667	84.1333	43.5	30.8000	19.5667	14.0000	11.4000	4.5667	16.0	59.7	
169	Uganda	155.7000	98.2333	47.3	29.6667	17.9000	14.5000	14.3000	5.3333	36.6	53.9	
43	Côte d'Ivoire	128.4333	86.0333	50.7	32.9667	23.4000	21.1000	20.1000	8.0667	23.9	51.8	
29	Cameroon	118.0333	79.1000	46.6	29.3000	18.8000	17.6000	16.5333	6.3000	24.4	46.6	

In [41]:
```
Filtered_DF6 = sortedDF.iloc[:,[0, -2, -1, -4, -3]]
display(Filtered_DF6.head())
```

	Country	Country Code	Continent	Population 2016 in millions	Suicide Rate per 100K
93	Lesotho	LSO	Africa	2.2	61.5
182	Zimbabwe	ZWE	Africa	16.0	59.7
169	Uganda	UGA	Africa	36.6	53.9
43	Côte d'Ivoire	CIV	Africa	23.9	51.8
29	Cameroon	CMR	Africa	24.4	46.6

Comment on previous cell output

The output displays a filtered dataframe that has all the necessary columns which will be utilised in order to visualise different trends of the global suicide rate.

Brief Explanation of following code cell

The cell below contains codes for a horizontal bar plot utilizing seaborn package, that shows top 10 countries with highest suicide rates per 100K population.

In [42]:
```
#Create a bar plot showing top 10 countries with highest suicide rate
```

```
def country_figure():
    sns.set_palette("RdPu_r", 14)
    top_10_countries = Filtered_DF6.head(10)
    plt.figure(figsize=(11,8))
    ax = sns.barplot(x="Suicide Rate per 100K", y="Country", data=top_10_countries, orie
    ax.set_ylabel("Country", fontsize = 20)
    ax.set_xlabel("Suicide Rate Per 100K", fontsize = 20)
    ax.set_title("10 Countries with Highest Suicide Rate per 100K Population", fontsize
    plt.xticks(fontsize=20)
    plt.yticks(fontsize=20)
    sns.despine()
    ax.grid(False)
    ax.tick_params(bottom=True, left=True)
    return(plt.show())
```

Brief Explanation of following code cell

The cell below contains codes for calculating total suicide rate per 100K for each continent in the year 2016. Thereafter, a code cell for vertical bar plot is included that shows the suicide rate distribution by continent.

In [43]:

```
continentList2 = Filtered_DF6.groupby(by = "Continent")['Suicide Rate per 100K'].sum().r
totalSuicide = Filtered_DF6.groupby(by='Continent')['Suicide Rate per 100K'].sum().sum()
percentage = ( continentList2["Suicide Rate per 100K"] / totalSuicide) * 100
continentList2["Overall Suicide Rate"] = percentage.round(1)
continentList2.drop("Suicide Rate per 100K", inplace=True, axis=1)
sortedContinentList = continentList2.sort_values(by = "Overall Suicide Rate", ascending
sortedContinentList
```

Out[43]:

	Continent	Overall Suicide Rate
0	Africa	43.7
2	Europe	23.3
1	Asia	16.9
3	North America	6.2
5	South America	5.9
4	Oceania	4.0

In [44]:

```
def continent_figure():
    sns.set_palette("RdPu_r", 1)
    plt.figure(figsize = (11,8))
    splot = sns.barplot(x = "Continent", y = "Overall Suicide Rate", data=sortedContinen
    for g in splot.patches:
        splot.annotate(format(g.get_height(), '.1f'),
                       (g.get_x() + g.get_width() / 2., g.get_height()),
                       ha = 'center', va = 'center', fontsize = 20, color = "black",
                       xytext = (0, 9),
                       textcoords = 'offset points')
    plt.xlabel("Continent Names", fontsize = 20)
    plt.ylabel("Suicide Rate %", fontsize = 20)
    plt.title("Global Percentage Suicide Rate per 100K Population, by Continent", fontsi
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
    sns.despine()
    return(plt.show())
```

Brief Explanation of following code cell

The following code cell contains codes for obtaining both population and total suicide rate per 100K for each continent grouping by "Continent" column and subsequently another code cell is added for plotting subplots of 2 pie charts using plotly package.

In [45]:
```
continentPopulation = Filtered_DF6.groupby(by = "Continent")['Population 2016 in million
continentPopulation["Suicide Rate per 100K"] = sortedContinentList["Overall Suicide Rate
continentPopulation
```

Out[45]:

	Continent	Population 2016 in millions	Suicide Rate per 100K
0	Africa	22.0	43.7
1	Asia	94.0	16.9
2	Europe	19.0	23.3
3	North America	27.0	6.2
4	Oceania	4.0	4.0
5	South America	35.0	5.9

In [46]:
```
def comparison_figure():
    labels = continentPopulation["Continent"]
    fig = make_subplots(1,2, specs=[[{'type':'domain'}, {'type':'domain'}]],
                        subplot_titles=['Continent Vs Population', 'Continent vs Suicide Rat
    fig.add_trace(go.Pie(labels=labels, values= continentPopulation["Population 2016 in 
                         name="Global Population in millions by Continent"), 1, 1)
    fig.add_trace(go.Pie(labels=labels, values= continentPopulation["Suicide Rate per 10
                         name="Global Suicide Rate per 100K by Continent"), 1, 2)
    fig.update_layout(title_text='Comparison between Population and Suicide Rate', font_
    return(fig.show())
```

Brief Explanation of following code cell

The code cell below contains all the codes required to show distribution of global suicide rate per 100K in the world map using choropleth map.

In [47]:
```
worldMap = Filtered_DF6.copy()
choroplethMap = go.Figure(
    data = {
        'type':'choropleth',
        'locations':worldMap['Country Code'],
        'locationmode':'ISO-3',
        'colorscale':'Portland',
        'z':worldMap['Suicide Rate per 100K'],
        'colorbar':{'title':'World Suicide Rate in 2016'},
        'marker': {
            'line': {
                'color':'rgb(255,255,255)',
                'width':2
            }
        }
    },
    layout = {
       'geo':{
            "showframe" : False,
            "showcoastlines" : False,
            "projection" :{
                "type" :'equirectangular'
        },
            'scope':'world',
```

```
        }
    })
```

Brief Explanation of following code cell

A fourth datset with information about different mental health facilities is uploaded and df.head() is called to show the first 5 columns of the dataframe.

In [48]:
```
facilitiesData = pd.read_csv('Facilities.csv')
display(facilitiesData.head())
```

	Country	Year	Mental _hospitals	health_units	outpatient _facilities	day _treatment	residential_facilities
0	Afghanistan	2016	0.003	0.012	0.006	NaN	NaN
1	Albania	2016	0.068	0.068	0.410	NaN	0.445
2	Algeria	2016	0.048	0.068	0.048	NaN	NaN
3	Angola	2016	0.011	NaN	NaN	NaN	0.014
4	Antigua and Barbuda	2016	1.001	NaN	NaN	NaN	NaN

Brief Explanation of following code cell

Occurence of NaN values is evident from the first 5 columns of this dataset, therefore, the number of null and duplicate values are first checked and then replaced by 0 for more accurate analysis.

In [49]:
```
duplicate = facilitiesData.duplicated().sum()
print("No of duplicates: ", duplicate)

nullValuesEachColumn_FD = facilitiesData.isnull().sum()  #counting number null values in
print("Null values for each column: ", "\n", nullValuesEachColumn_FD)

totalNullValues_FD = facilitiesData.isnull().sum().sum()  #counting number of null value
print("Null values in entire df before cleaning: ", totalNullValues_FD)
```

```
No of duplicates:  0
Null values for each column: 
 Country                    0
Year                       0
Mental _hospitals         22
health_units              10
outpatient _facilities    12
day _treatment            61
residential_facilities    67
dtype: int64
Null values in entire df before cleaning:  172
```

In [50]:
```
facilitiesData.fillna(0.0, inplace = True)  #replacing the null values with 0.0

totalNullValues_FD2 = facilitiesData.isnull().sum().sum() #counting number of null value
print("Null values in entire df after cleaning: ", totalNullValues_FD2)
```

```
Null values in entire df after cleaning:  0
```

Brief Explanation of following code cell

The total number of facilities per 100K population available in the mental health sector of each country is calculated and the corresponding values are assigned to a column "Total Facilities per 100K". Then this column is normalised using the min-max scaling approach and the values are allocated to the "Normalised Scores of Total Facilities" column in the dataframe.

In [51]:
```
f = facilitiesData.copy()
f['Total Facilities per 100K'] = f.iloc[:,2:-1].mean(axis=1).round(2)
#NORMALIZING THE 'Total facilties per 0.1 million' column using min-max scaling approach
totalFacilities = 'Total Facilities per 100K'
v = totalFacilities
f['Normalized Scores of Total Facilities'] = ((f[v] - f[v].min()) / (f[v].max() - f[v].m
#Sorting the data by Normalized Scores of Total Facilities
f.sort_values(by='Normalized Scores of Total Facilities', ascending = False, inplace = T
f.head()
```

Out[51]:

	Country	Year	Mental _hospitals	health_units	outpatient _facilities	day _treatment	residential_facilities	Total Facilities per 100K	Normalized Scores of Total Facilities
32	Estonia	2016	0.152	0.684	14.820	17.176	3.496	8.21	1.00
84	Saint Lucia	2016	0.564	0.000	19.751	0.000	0.000	5.08	0.62
50	Japan	2016	8.314	0.450	7.223	3.759	0.366	4.94	0.60
85	Samoa	2016	0.516	0.516	12.387	0.516	4.645	3.48	0.42
61	Monaco	2016	0.000	2.610	5.221	5.221	0.000	3.26	0.40

Comment on previous cell output

This resulting dataframe gives the 5 countries with highest facilities in the mental heath sector.

Brief Explanation of following code cell

For all these countries, their corresponding continent names are derived using the pre-defined function "get_continent" and the continent names are added in a separate column of the dataframe.

In [52]:
```
f["Country"] = f["Country"].replace(replace_values)
f_country_list = list(f["Country"])
f_continent_list = [get_continent(country) for country in f_country_list]

# addition of a continent column with continent names
f['Continent'] = f_continent_list
f["Continent"] = f["Continent"].replace(continent_name)
f.head()
```

Out[52]:

	Country	Year	Mental _hospitals	health_units	outpatient _facilities	day _treatment	residential_facilities	Total Facilities per 100K	Normalized Scores of Total Facilities	C
32	Estonia	2016	0.152	0.684	14.820	17.176	3.496	8.21	1.00	
84	Saint Lucia	2016	0.564	0.000	19.751	0.000	0.000	5.08	0.62	
50	Japan	2016	8.314	0.450	7.223	3.759	0.366	4.94	0.60	
85	Samoa	2016	0.516	0.516	12.387	0.516	4.645	3.48	0.42	

61	Monaco	2016	0.000	2.610	5.221	5.221	0.000	3.26	0.40

Brief Explanation of following code cell

The below code cell represents the distribution of different facilities available as a percentage, across all continents.

In [53]:
```
Filtered_f = f[['Total Facilities per 100K', 'Continent']]
f1 = Filtered_f.groupby(by='Continent')['Total Facilities per 100K'].sum()
f1_total = Filtered_f.groupby(by='Continent')['Total Facilities per 100K'].sum().sum()
percentage_TotalFac = ( f1 / f1_total) * 100
percentage_TotalFac.sort_values(ascending = False, inplace=True)

percentage_TotalFac
```

Out[53]:
```
Continent
Europe           37.2084
Asia             19.4241
North America    17.3843
Oceania          14.6914
Africa            5.7326
South America     5.5593
Name: Total Facilities per 100K, dtype: float64
```

Brief Explanation of following code cell

In the code cell below, our final dataset is uploaded that gives insights on different human resources available in each country towards the mental health sector. The df.head() is called to get an overview of the dataset.

In [54]:
```
HRData = pd.read_csv("Human Resources.csv")
display(HRData.head())
```

	Country	Year	Psychiatrists	Nurses	Social_workers	Psychologists
0	Afghanistan	2016	0.231	0.098	NaN	0.296
1	Albania	2016	1.471	6.876	1.060	1.231
2	Angola	2016	0.057	0.660	0.022	0.179
3	Antigua and Barbuda	2016	1.001	7.005	4.003	NaN
4	Argentina	2016	21.705	NaN	NaN	222.572

Brief Explanation of following code cell

Similar to the fourth dataset, the data quality issues in the HRData dataframe are taken into considersation and similar measures are taken to cater the issue.

In [55]:
```
duplicateHR = HRData.duplicated().sum()
print("No of duplicates: ", duplicateHR)
nullValuesInEachColumn_HRD = HRData.isnull().sum()  #counting number null values in each
print("Null values for each column: ", "\n", nullValuesInEachColumn_HRD)
totalNullValues_HRD = HRData.isnull().sum().sum()#counting number of null values in enti
print("Null values in entire df before cleaning: ", totalNullValues_HRD)
```

```
No of duplicates:  0
Null values for each column: 
 Country          0
```

```
Year                0
Psychiatrists       3
Nurses             16
Social_workers     39
Psychologists      23
dtype: int64
Null values in entire df before cleaning:  81
```

In [56]:

```python
HRData.fillna(0.0, inplace = True) #replacing the null values with 0.0

totalNullValues_HRD2 = HRData.isnull().sum().sum()#counting number of null values in ent
print("Null values in entire df after cleaning: ", totalNullValues_HRD2)
```

```
Null values in entire df after cleaning:  0
```

Brief Explanation of following code cell

The total number of human resources per 100K population available in the mental health sector of each country is evaluated and the corresponding values are assigned to a column "Total HR per 100K". Keeping similarity with the previous dataset, this column is normalised using a min-max scaling approach, and the values are added to the "Normalised Scores of Total HR" column in the dataframe.

In [57]:

```python
h = HRData.copy()
h['Total HR per 100K'] = h.iloc[:,2:-1].mean(axis=1).round(2)
#NORMALIZING THE 'Total facilties per 0.1 million' column using min-max scaling approach
totalHR = 'Total HR per 100K'
c = totalHR
h['Normalized Scores of Total HR'] = ((h[c] - h[c].min()) / (h[c].max() - h[c].min())).r
#Sorting the data by Normalized Scores of Total Facilities
h.sort_values(by='Normalized Scores of Total HR', ascending = False, inplace = True)
h.head()
```

Out[57]:

	Country	Year	Psychiatrists	Nurses	Social_workers	Psychologists	Total HR per 100K	Normalized Scores of Total HR
58	Monaco	2016	31.326	83.536	102.592	53.515	72.48	1.00
96	Turkey	2016	1.637	150.251	1.643	2.537	51.18	0.71
67	New Zealand	2016	28.540	75.132	0.000	0.000	34.56	0.48
48	Japan	2016	11.867	83.805	8.328	3.037	34.67	0.48
22	Costa Rica	2016	3.931	5.699	76.957	142.018	28.86	0.40

Comment on previous cell output

This resulting dataframe gives the 5 countries with highest HR available in the mental heath sector.

Brief Explanation of following code cell

The continent names for all these corresponding countries are derived using the pre-defined function "get_continent" and the output is allocated in a separate column of the dataframe.

In [58]:

```python
h["Country"] = h["Country"].replace(replace_values)
h_country_list = list(h["Country"])
h_continent_list = [get_continent(country) for country in h_country_list]

# addition of a continent column with continent names
h['Continent'] = h_continent_list
```

```
h["Continent"] = h["Continent"].replace(continent_name)
h.head()
```

Out[58]:

	Country	Year	Psychiatrists	Nurses	Social_workers	Psychologists	Total HR per 100K	Normalized Scores of Total HR	Continent
58	Monaco	2016	31.326	83.536	102.592	53.515	72.48	1.00	Europe
96	Turkey	2016	1.637	150.251	1.643	2.537	51.18	0.71	Asia
67	New Zealand	2016	28.540	75.132	0.000	0.000	34.56	0.48	Oceania
48	Japan	2016	11.867	83.805	8.328	3.037	34.67	0.48	Asia
22	Costa Rica	2016	3.931	5.699	76.957	142.018	28.86	0.40	North America

Brief Explanation of following code cell

The code below represents the distribution of different human resources available as a percentage, across all continents.

In [59]:

```
Filtered_h = h[['Total HR per 100K', 'Continent']]
h1 = Filtered_h.groupby(by='Continent')['Total HR per 100K'].sum()
h1_total = Filtered_h.groupby(by='Continent')['Total HR per 100K'].sum().sum()
percentage_TotalHR = ( h1 / h1_total) * 100
percentage_TotalHR.sort_values(ascending = False, inplace=True)

percentage_TotalHR
```

Out[59]:

```
Continent
Europe           41.4314
Asia             24.2959
North America    14.9213
Oceania           9.4973
South America     6.4276
Africa            3.4265
Name: Total HR per 100K, dtype: float64
```

Brief Explanation of following code cell

Next, we are combining the required columns from 3 of our dataframes to create a merged dataframe.

In [60]:

```
#Merging all 3 DF's based on common Country
HRData_cp = h.iloc[:,[0,6]]
facilitiesData_cp = f.iloc[:,[0,7]]
merge1 = pd.merge(HRData_cp, facilitiesData_cp, on='Country')
display(merge1.head())
```

	Country	Total HR per 100K	Total Facilities per 100K
0	Monaco	72.48	3.26
1	Turkey	51.18	0.16
2	New Zealand	34.56	0.15
3	Japan	34.67	4.94
4	Costa Rica	28.86	0.48

In [61]:

```
suicideData_cp = Filtered_DF6.iloc[:,[0, 3, 4]]
```

```
merge2 = pd.merge(merge1, suicideData_cp, on= "Country")
display(merge2.head())
#print("Shape of merge2: ", merge2.shape)
```

	Country	Total HR per 100K	Total Facilities per 100K	Population 2016 in millions	Suicide Rate per 100K
0	Turkey	51.18	0.16	79.5	8.5
1	New Zealand	34.56	0.15	4.7	14.8
2	Japan	34.67	4.94	125.3	22.6
3	Costa Rica	28.86	0.48	4.9	10.0
4	United States of America	25.05	0.23	323.9	20.5

Comment on previous cell output

This merged dataframe will form the basis for a majority of visualisations implemented for this objective.

Brief Explanation of following code cell

A correlation matrix is derived betwen several column variables of finalMerge dataframe.

In [62]:
```
finalMerge = merge2.copy()
corrDF_cp = (finalMerge)[['Country', 'Population 2016 in millions','Total HR per 100K',
                          'Total Facilities per 100K', 'Suicide Rate per 100K']]
corrDF = corrDF_cp.copy()
#checking for null values
null = corrDF.isnull().sum()
print("Null :", "\n", null)
#checking for duplicates
dup = corrDF.duplicated().sum()
print("Duplicate: ", dup)
```

```
Null :
 Country                         0
Population 2016 in millions    0
Total HR per 100K              0
Total Facilities per 100K      0
Suicide Rate per 100K          0
dtype: int64
Duplicate:  0
```

Brief Explanation of following code cell

The cell below has codes to plot 2 subplots showing top 10 countries with highest facilities and human resources.

In [63]:
```
def barGraph():
    plt.rcParams["figure.figsize"] = (12,6)
    #Country vs Facilities
    top10Countries_FD = f.head(10)
    ax= plt.subplot(1, 2, 1)
    top10Countries_FD[::-1].plot.barh(x='Country', y='Normalized Scores of Total Facilit
    # set the label
    plt.title('TOP 10 countries with Maximum Facilities', fontsize=20, color = "Blue")
    plt.xlabel('Normalized Scores of Total Facilities', fontsize=20)
    plt.xticks(fontsize=15)
    plt.ylabel('Country', fontsize=20)
    plt.yticks(fontsize=15)
```

```
    sns.despine(bottom=True)
    ax.grid(False)
    ax.tick_params(bottom=False, left=True)
    ax.get_legend().remove()
    #plt.legend(['Normalized Scores of Total Facilities'], loc='right', prop={'size': 10
    #Country vs HR
    top10Countries_HRD = h.head(10)
    ax= plt.subplot(1, 2, 2)
    top10Countries_HRD[::-1].plot.barh(x='Country', y='Normalized Scores of Total HR', a
    plt.title('TOP 10 countries with Maximum HR Resources', fontsize=20, color = "Blue")
    plt.xlabel('Normalized Scores of HR Resources', fontsize=20)
    plt.xticks(fontsize=15)
    plt.ylabel('Country', fontsize=20)
    plt.yticks(fontsize=15)
    sns.despine(bottom=True)
    ax.grid(False)
    ax.tick_params(bottom=False, left=True)
    ax.get_legend().remove()
    #plt.legend(['Normalized Scores of HR Resources'], loc='right', prop={'size': 10})
    plt.tight_layout()
    return(plt.show())
```

Brief Explanation of following code cell

The cell below has codes to pictorially depict 2 pie charts elucidating the distribution of facilities and human resources for the country having the highest number of them.

In [64]:
```
def pieChart():
    #Country vs Facilities
    top10Countries_FD = f.head(10)
    changedIndex_FD = top10Countries_FD.set_index('Country')
    fig, axes = plt.subplots(1, 2, figsize=(15, 20))
    ax1 = plt.subplot(121, aspect='equal')
    colors1 = ['#FF4040','#00CED1', '#B7C3F3', '#DD7596', '#8EB897']
    ax1.pie(changedIndex_FD.iloc[0,[1,3,2,4,5]], labels=changedIndex_FD.iloc[0,[1,3,2,4,
            autopct='%1.0f%%', colors = colors1, textprops={'fontsize': 15})
    ax1.set_title(changedIndex_FD.index[0], fontsize=30, color = "Blue")
    #Country vs HR
    top10Countries_HRD = h.head(10)
    changedIndex_HRD = top10Countries_HRD.set_index('Country')
    ax2 = plt.subplot(122, aspect='equal')
    ax2.pie(changedIndex_HRD.iloc[0,1:5], labels=changedIndex_HRD.iloc[0,1:5].index, sta
            autopct='%1.0f%%', colors = colors1, textprops={'fontsize': 15})
    ax2.set_title(changedIndex_HRD.index[0],  fontsize=30, color = "Blue")
    fig.subplots_adjust(wspace=1)
```

Brief Explanation of following code cell

The cell below contains the codes to plot 2 line charts in one graph, showing the distribution of facilities and human resources as a percentage across all continents.

In [65]:
```
def lineGraphContinentalHRFac():
    plt.figure(figsize=(18,6))
    plt.plot(percentage_TotalHR, marker='*', label='Human Resources', color="c")
    plt.plot(percentage_TotalFac, marker='*', label='Facilities', color="m")
    plt.legend()
    plt.xlabel('Continents', fontdict = {'fontsize' : 15})
    plt.ylabel('Percentage', fontdict = {'fontsize' : 15})
    plt.title('Continent wise trends between HR and Facilities',fontdict = {'fontsize' :
    plt.show()
```

Brief Explanation of following code cell

Illustrated below is the code to plot hatmap to show correlation between column variables using seaborn package.

In [66]:
```
def corrMatrix():
        hm = sns.heatmap((corrDF).corr(), annot = True)
        hm.set(title = "Correlation matrix of Mental Health\n")
        return(plt.show())
```

Project Outcome (10 + 10 marks)

Overview of Results

Suicide is an extremely tragic issue that causes pain to hundreds of thousands of people every year around the world. According to the World Health Organization (WHO) and the Global Burden of Disease study estimation, every 40 seconds an individual commits suicide. This intrigued us to explore in detail the global distribution of suicide and investigate the trend of suicide rates in different demographic information available in the dataset. We used a systematic approach to display countries and continents with the highest rate of suicide and monitor the inclination in different genders and age groups. The risk of suicide has been estimated to be 5–8% for several mental disorders, such as depression, alcoholism, and schizophrenia. Hence, we wanted to investigate if there is any correlation between different facilities and resources available in the mental health sector of a country and its prevailing suicide rate. The objective of this data analysis project is to contribute to an informed, open debate about ways to prevent suicide.

Objective 1 : Year-wise trends of suicide rates across 6 continents.

Explanation of Results

For our first objective, we used the Age Standardized Suicide rate dataset wherein we were interested in exploring the change in global suicide year across 4 years-2000, 2010, 2015, and 2016. By evaluating and plotting the overall suicide rate for each year in Figure 1, we discovered that the overall suicide rate was differing between the early 21st century to the current years and in the process, found out that the suicide rate has drastically decreased over these years. Then, we showed the suicide rate distribution among the three sexes namely male, female, and both sexes for all 4 years, as evident in Figure 2. From this figure, we deciphered that the suicide rate in males (as represented by the green bar) remained high throughout the years and is nearly three times as high as compared to females. Later, we went ahead and wanted to group the countries to their respective continents to observe the suicide rate trend across the 6 continents throughout the years in search of any interesting patterns that the data might reveal. Using matplotlib, we plotted the suicide rates over the continents across all the years in Figure 3 and found out that Africa had the highest suicide rate for all four years followed by Europe and Asia, whereas, Oceania consistently had the least suicide rate throughout.

Visualisation

- *Figure 1: The line plot gives a clear distribution of global suicide rate for 4 years.*
- *Figure 2: The grouped bar plot gives a vivid representation of overall suicide rate across 3 genders.*

- *Figure 3: The stacked line chart depicts change of suicide rate across 6 continents for all 4 years.*

Figure 1: Yearly Suicide Rate

In [67]:
```
lineGraphYearlySuicideRate()
```

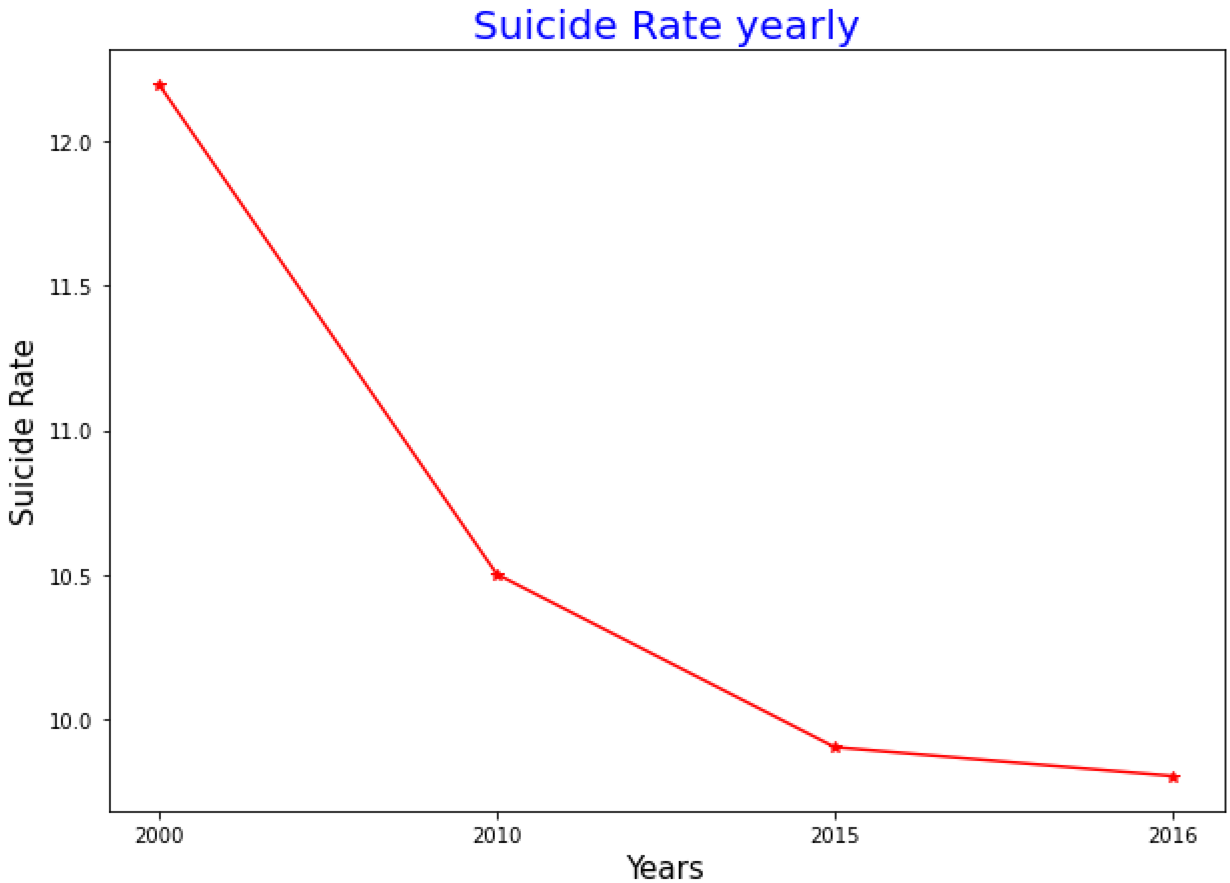


Figure 2: Gender-wise Suicide Rate Trend Across Years

In [68]:
```
barGraphGenderSuicideRate(ageSuicideDF)
```

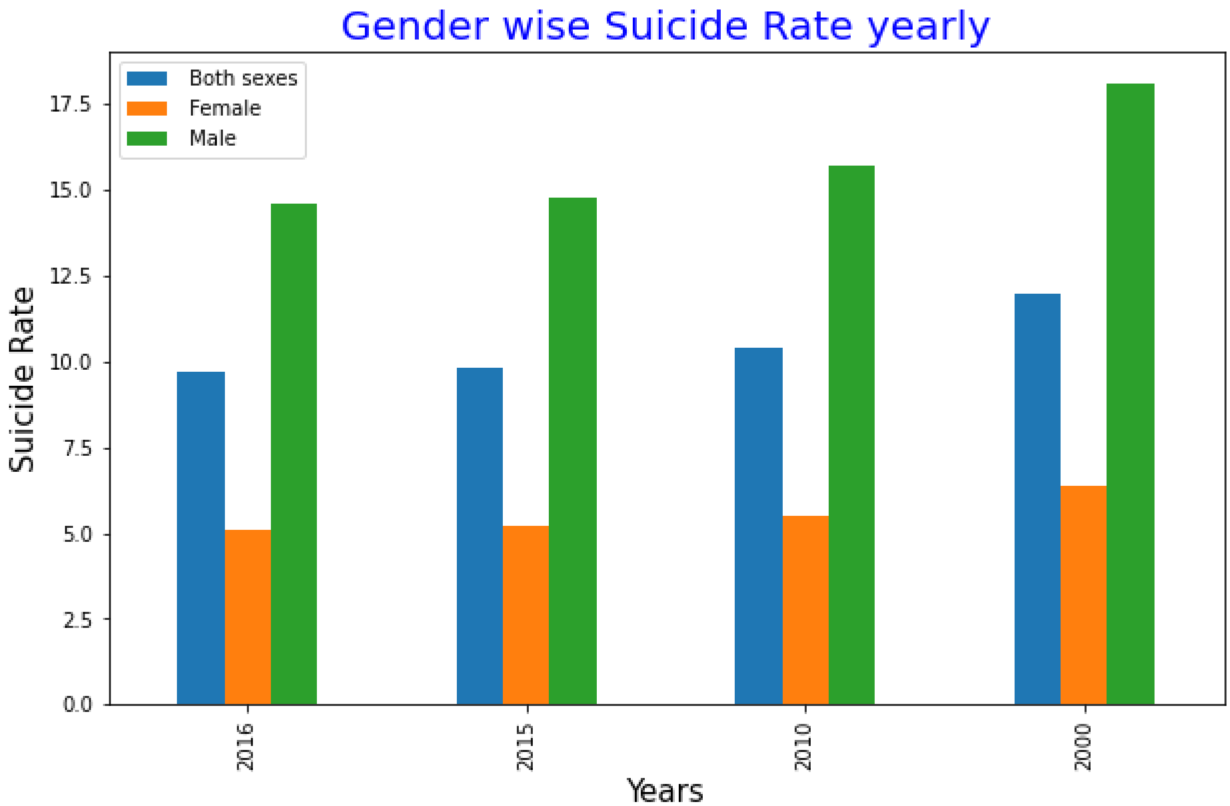
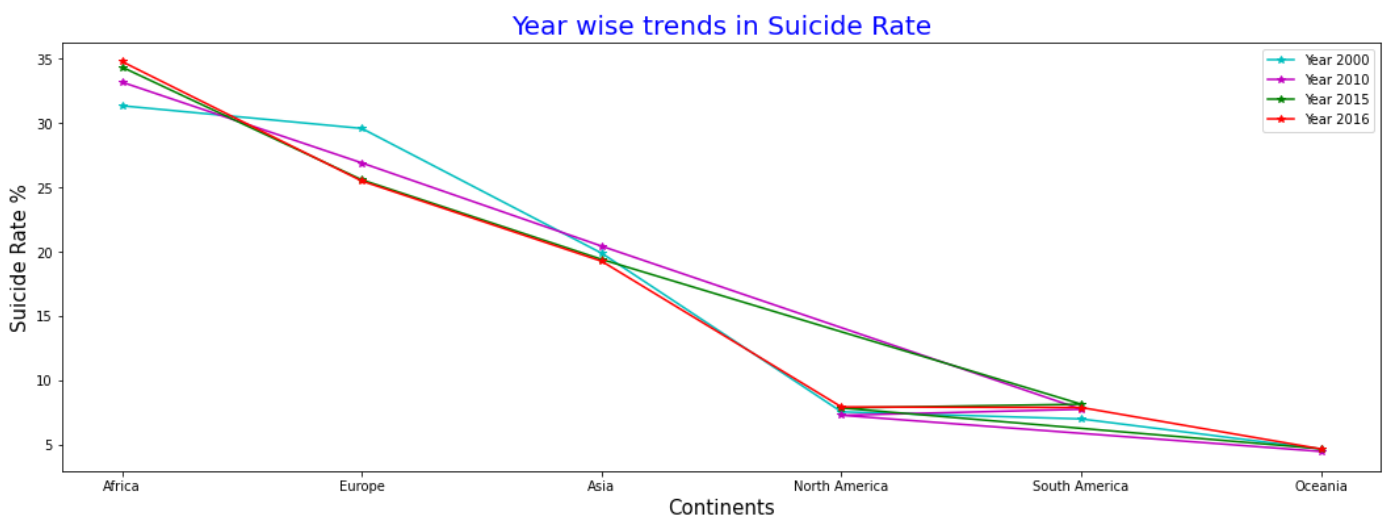


Figure 3: Year-wise Suicide Rate Trend Across Continents

In [69]:
```
lineGraphContinentalSuicideRate()
```



Objective 2: Suicide rate in different genders and age groups.

Explanation of Results

We have used the Crude Suicide Rate dataset that comprises of suicide rates of 183 countries worldwide divided among the three genders and eight different age groups ranging from 10 years to 80 above age bands. We cleaned the dataset using Pandas and observed for any unusual outliers or duplicate values. To achieve uniformity and to understand the trends of the suicide rate among genders across the world, we have used the min-max normalization that allowed us to plot the trends on a uniform scale. According to the dataset, we have 3 genders including both male, female as well as both sexes, and based on our analysis we can confirm that globally, suicide rates in men are highest in comparison to the other 2 genders, as represented in Figure 4. Furthermore, we were also keen to understand how the suicide rate is varying among eight different age groups in each gender. The dataset consisted of columns of each age group which were grouped by each gender. The results were quite astonishing as the trends depicted that males between the age of 60 to 69 have had the highest suicide rate across 183 countries, whereas for females, the age group 30 to 39 are more prone to commit suicide. The suicide rate in both sexes fairly remained the same as that of male with the highest suicide rate for the age band 60 to 69. Through the analysis coming from Figure 5-7, the trends seemed to be quite surprising and away from the misconception that most suicides are committed by teenagers or by people in their mid-30s.

Visualisation

- *Figure 4: The donut plot reflects worldwide distribution of suicide rate by genders.*
- *Figure 5: The blue line plot depicts global suicide rate change for different age groups, by male.*
- *Figure 6: The magenta line plot depicts global suicide rate change for different age groups, by female.*
- *Figure 7: The green line plot depicts global suicide rate change for different age groups, by both sexes.*

Figure 4: Gender vs Suicide Rate Worldwide in 2016

In [70]:
```
donutChartGender()
```

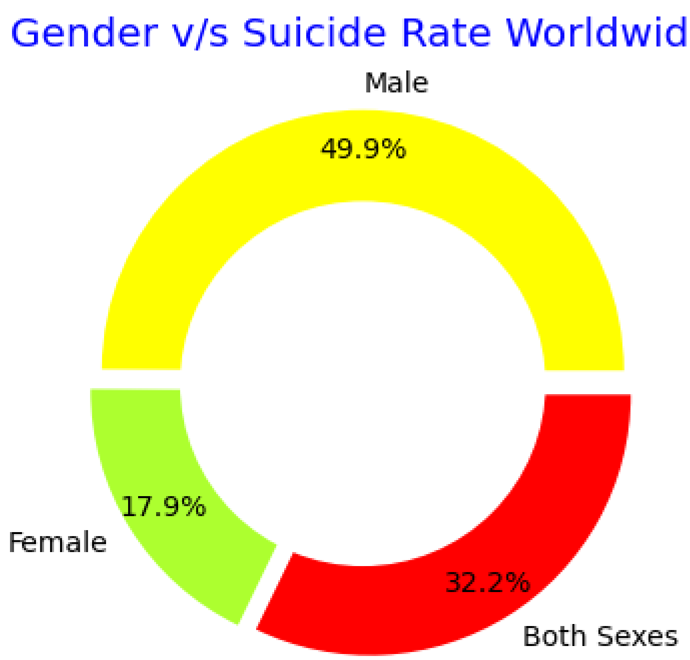


Figure 5: Male suicide rate age wise globally

In [71]:
```
lineGraphMale()
```

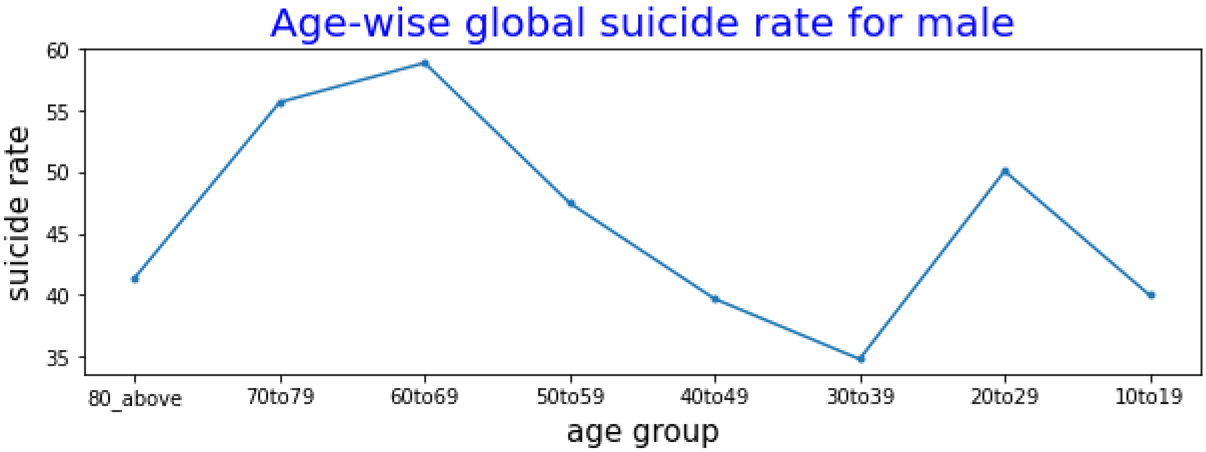


Figure 6: Female suicide rate age wise globally

In [72]:
```
lineGraphFemale()
```

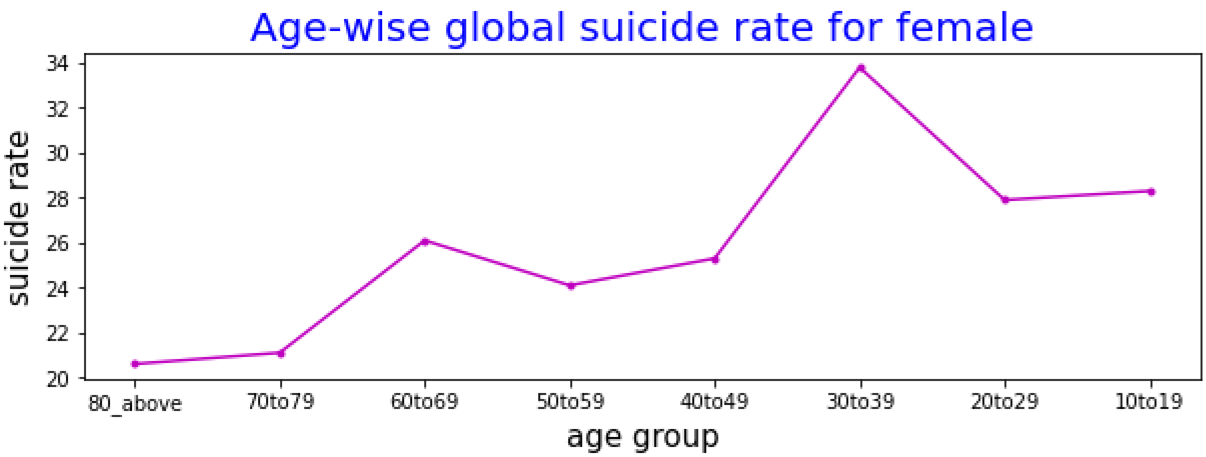
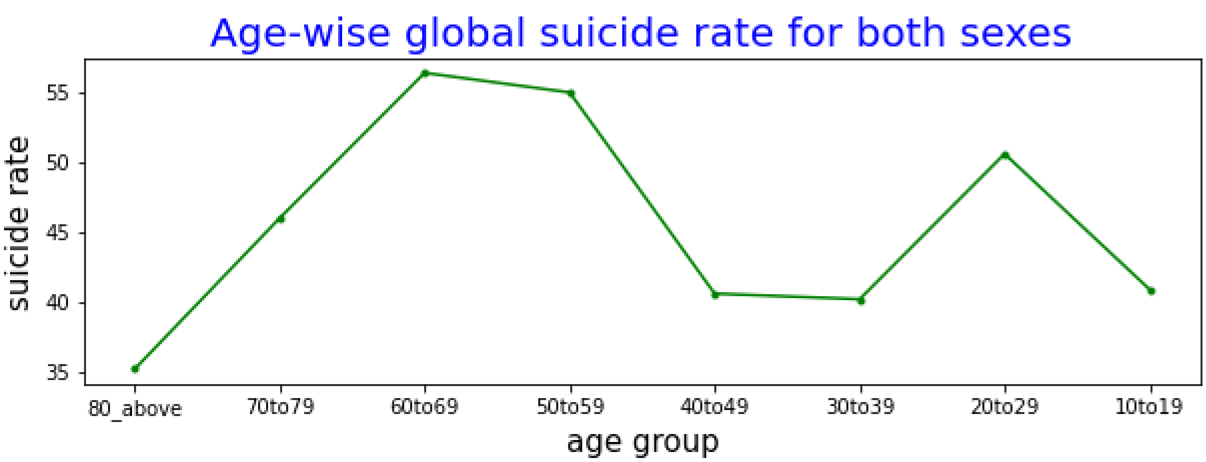


Figure 7: Both sexes suicide rate age wise globally

In [73]:
```
lineGraphBothSexes()
```



Objective 3: World map representation showing dispersal of the suicide rate.

Explanation of Results

Through this objective, we wanted to depict the complete distribution of suicide rates across 183 countries of the world in the year 2016. To achieve this, we evaluated the mean suicide rate for each country combining the 3 genders and 8 age bands to obtain the overall suicide rate for that respective country. By using Seaborn, we plotted a horizontal bar plot showing the top 10 countries with the highest number of suicides (Figure 8) and deduced that Lesotho was the highest with a suicide rate of 61.5, while in contrast, Antigua and Barbuda had the lowest suicide rate of the value 1.1 (Table 1). Following this, we displayed the overall suicide rate across all continents in Figure 9 and perceived that Africa's suicide rate is close to two times that of Europe and over ten times higher than that of Oceania. We were curious to investigate the trend of suicide rates across all continents in comparison with the population of that corresponding continent (Figure 9) to discover if there is any underyling causal relationship among the two parametes. In the pie plots, we observed even though Asia occupies nearly 50% of the world population, the suicide rate per 100 K is around 17%. In contrast to this, Africa, despite being the 4th largest continent by population, holds the highest number of suicides rate of 43.7%. Finally, to give a complete picture of countries with their corresponding suicide rates, we plotted a Choropleth map that vividly demonstrates the distribution of the global suicide rate.

Table-1:

In [74]:
```
display(Filtered_DF6.head(1))
display(Filtered_DF6.tail(1))
```

	Country	Country Code	Continent	Population 2016 in millions	Suicide Rate per 100K
93	Lesotho	LSO	Africa	2.2	61.5

	Country	Country Code	Continent	Population 2016 in millions	Suicide Rate per 100K
4	Antigua and Barbuda	ATG	North America	0.09	1.1

Visualisation

- *Figure 8: The horizontal bar plot reflects 10 countries with highest suicide rates.*
- *Figure 9: The vertical bar plot illustrates descending order ofsuicide rate distribution across continents.*
- *Figure 10: The pie subplots shows comparison between population and suivide rate across continents.*
- *Figure 11: The choropleth map depicts distribution of global suicide rate on a world map.*

Figure 8: Global Suicide Rate, by Country

In [75]:
```
country_figure()
```

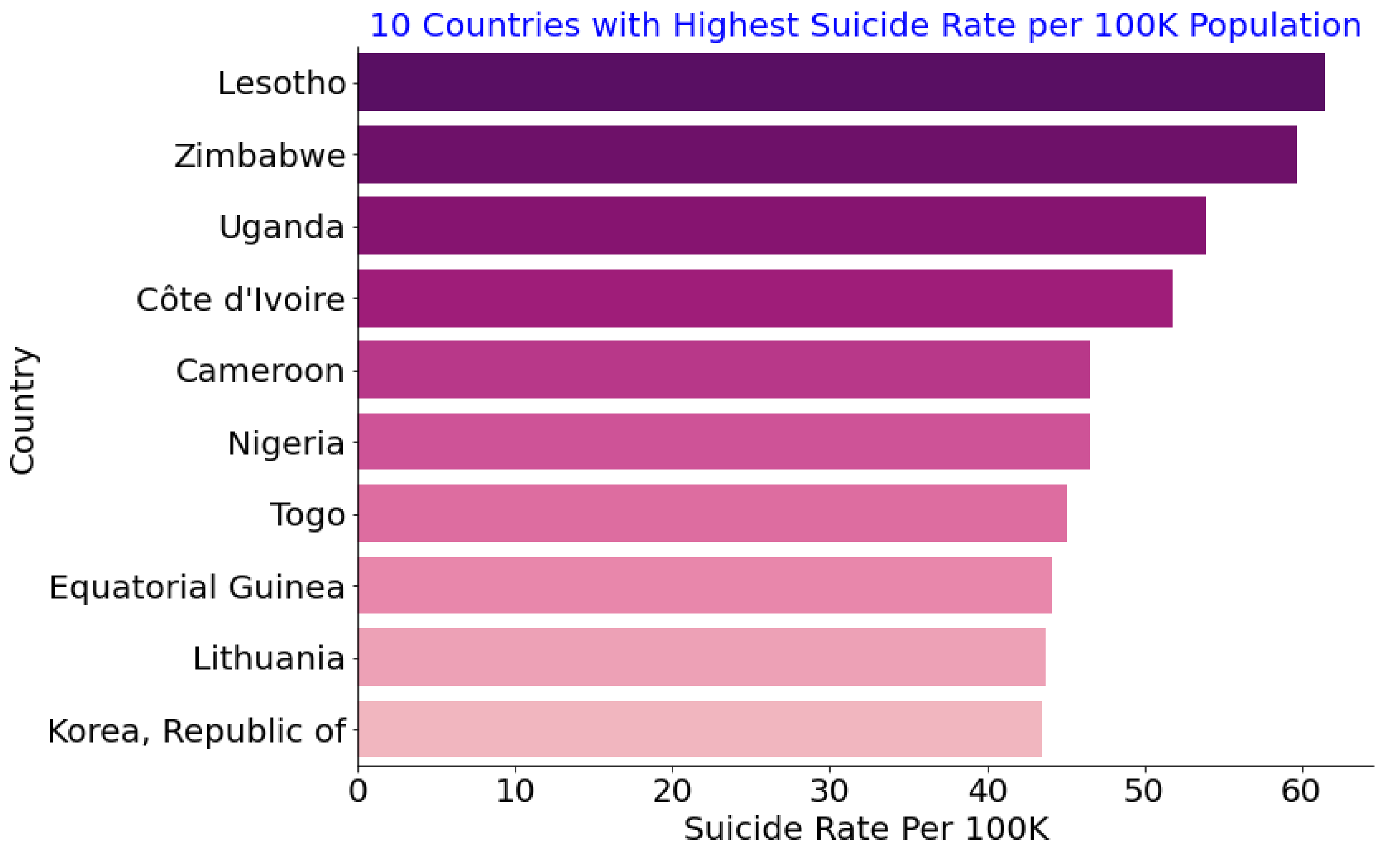


Figure 9: Global Suicide Rate, by Continent

In [76]:
```
continent_figure()
```

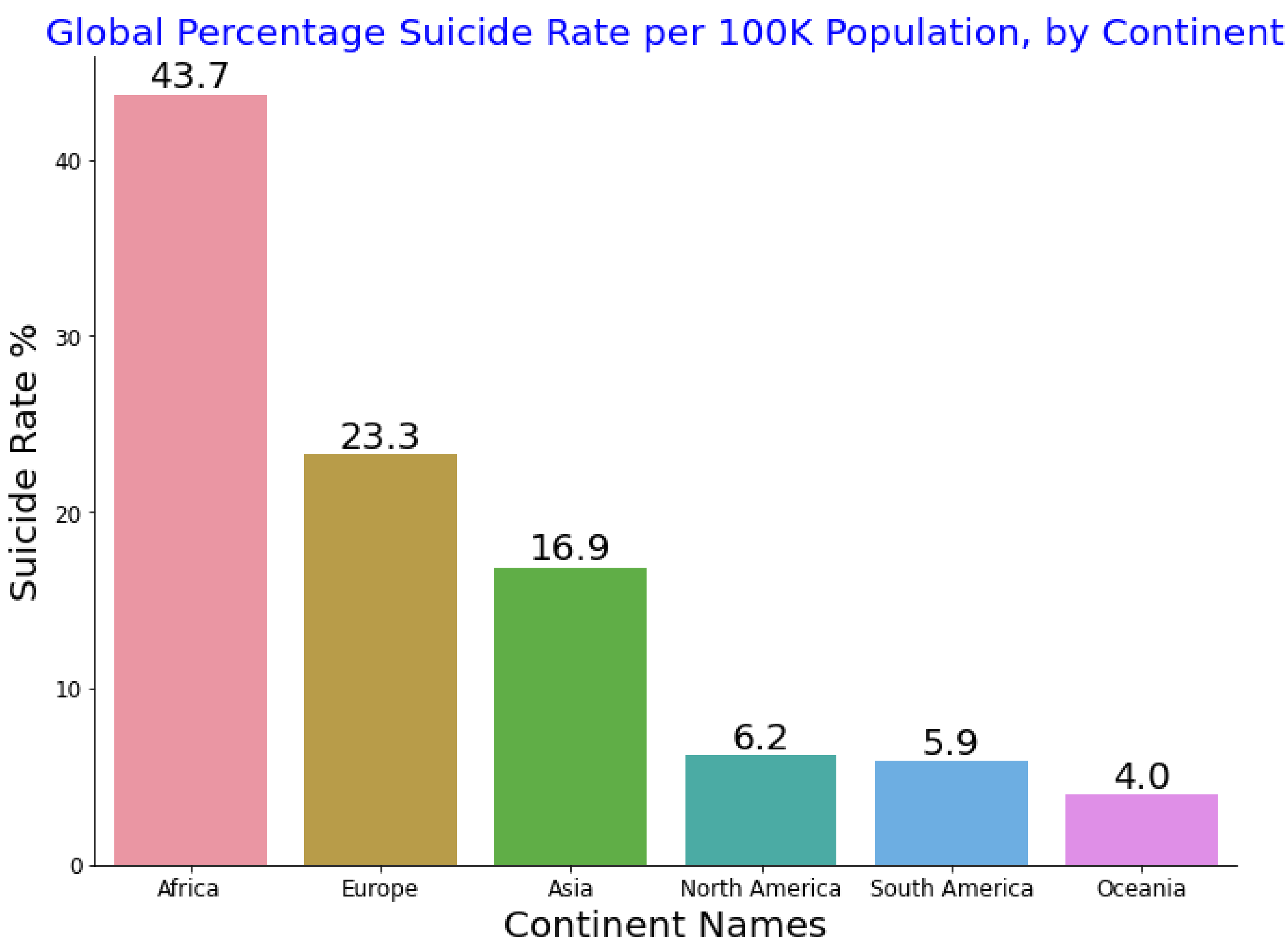


Figure 10: Comparison of Continent Population vs Suicide Rate

In [77]:
```
comparison_figure()
```

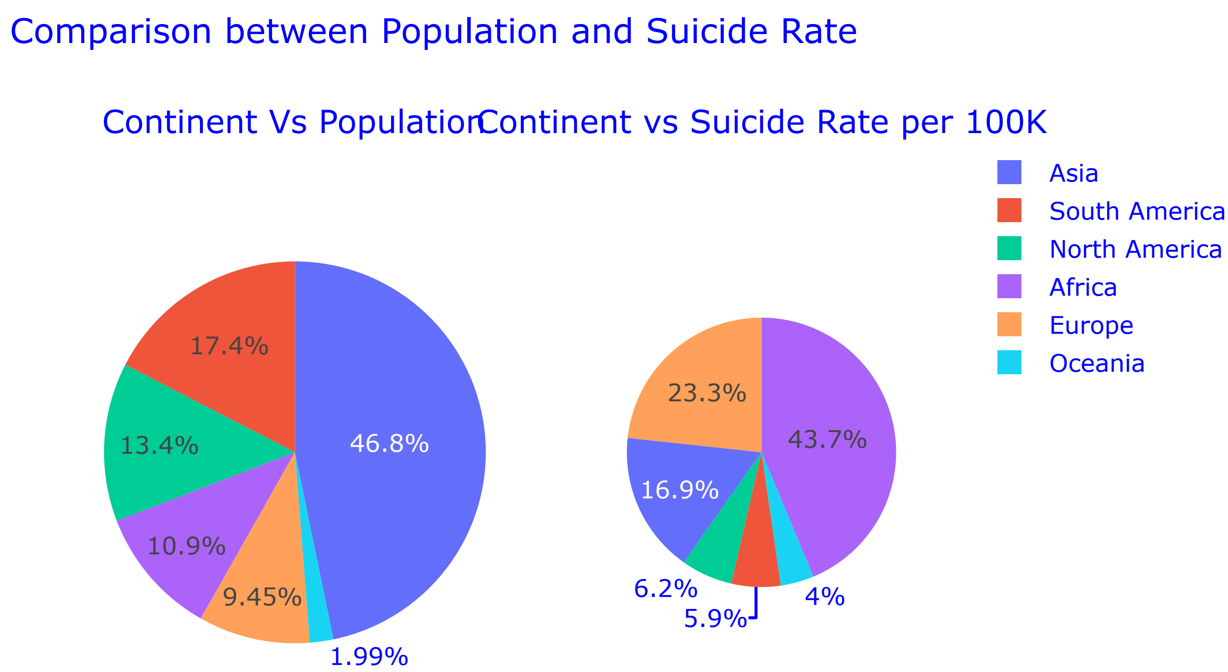
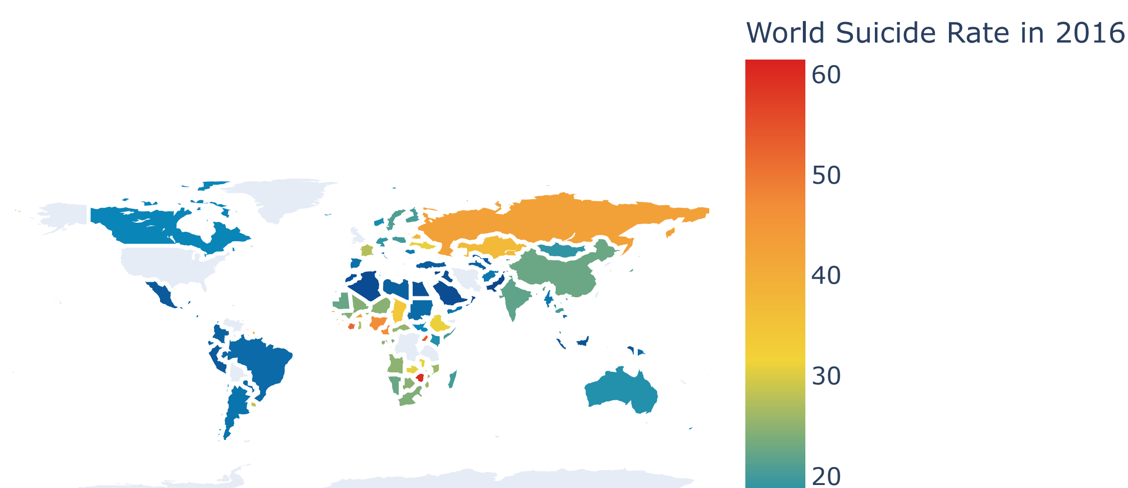
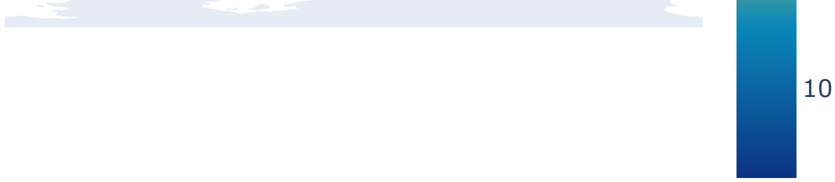


Figure 11: World Suicide Rate in 2016

In [78]:
```
choroplethMap
```





Objective 4: Distribution of Human Resource and Mental Health Facilities across Countries.

Explanation of Results

We have used the datasets designated for human resources (HR) and mental health facilities (Facility) collaboratively to acquire the results for this objective. The datasets hold the number of Health and Human Resource units per every 100K population of each country respectively. It is worth mentioning that HR/Facility units are categorized into 4 sub-domains based on their types. Initially, we cleaned the datasets using Pandas and then aggregated the HR/Facility units across each of their categories to obtain the Total units per every 100K population for each country. Furthermore, we normalized the values thus obtained using the Min-Max Scaling approach to ensure that their spread is uniform and accordingly sorted them in their decreasing order. For ease of comparison, we chose to limit our analysis within each dataset to only the top 10 countries having the highest Total units per every 100K population. We used horizontal bar graphs to visualize the trends in values across the top 10 countries (Figure 12). Through this plot, we observed that Monaco has the highest number of human resources and comes 5th in the list of top 10 countries for mental health facilities, where Estonia holds the first position. Then we employed a pie chart (Figure 13) to portray the specific breakdown for the HR categories, namely nurses, psychiatrists, psychologists, and social workers, whereas the categories for the Facility dataframe include mental hospitals, resident_facilities, day treatment, health units, and outpatient facilities. Thereafter, in Figure 14, we illustrate how these 2 factors (facility and HR units) are dispersed across all continents. From the findings of this graph, we can draw the inference that Africa's highest suicide rate is in consequence of the lowest number of facilities and human resources available in the mental health sector of the countries included in the continent. We could also decipher that Europe, despite ranking highest in its mental health resources, has the second-largest suicide rate in the world. This led us to anticipate that a wider population of Europe is either unaware of the mental health resources or is not comfortable accessing them.

Post gathering all plausible inferences from each of our objectives we felt the need to go one step ahead and understand how each of our primal parameters depends on the other. By employing functionalities within seaborn, we were able to design a correlation matrix which depicted how closely the parameters – Health Facilities, Human Resources, and Suicide Rates were related to each other (Figure 15). We only found a weak correlation between Total HR per 100K and Total Facilities per 100K.

Visualisation

- *Figure 12: The 2 horizontal bar plot helps in visualising 10 countries with maximum facilities(left plot) and human resources (right plot) respectively.*
- *Figure 13: The 2 pie plots illustrates different categories of facilities and human resources in mental health sector.*

- *Figure 14: The line graph signifies percentage of Human Resources and Facilities between continents.*
- *Figure 15: The heatmap signifies a correlation matrix between various column variables of the combined dataset.*

Figure 12: Top 10 Countries with Maximum Facilities and Maximum Human Resources.

In [79]:
```
barGraph()
```

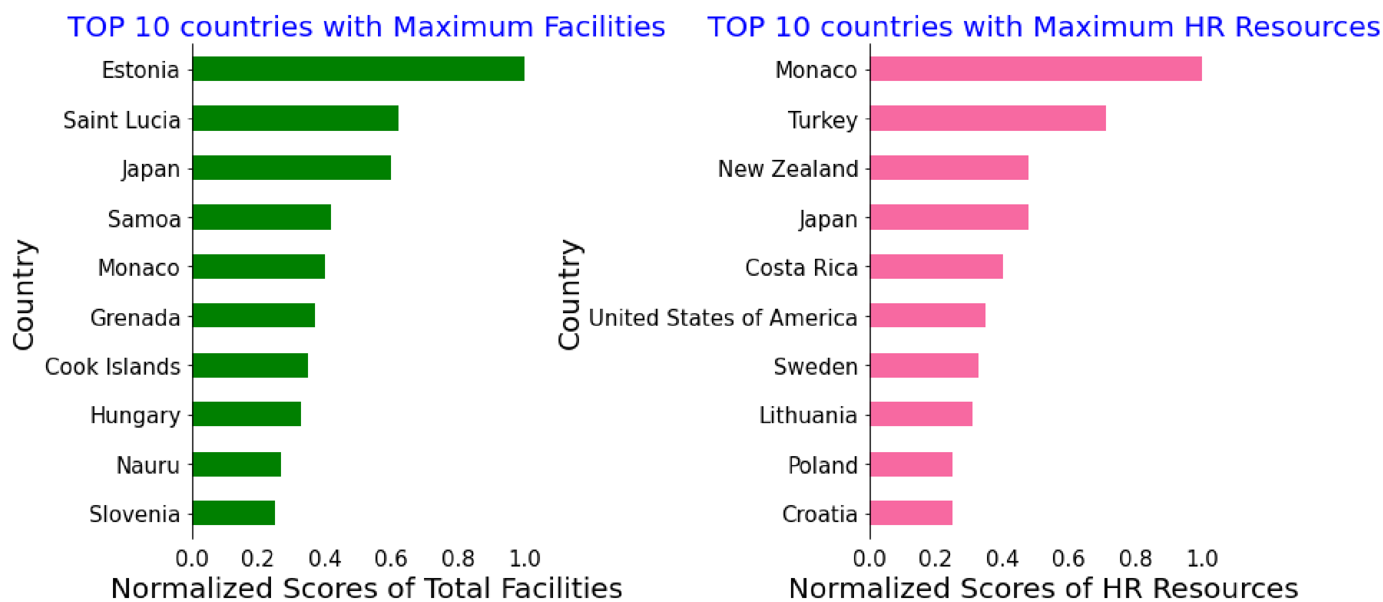


Figure 13: Various Facilities and Human Resources in Mental Health Sector

In [80]:
```
pieChart()
```

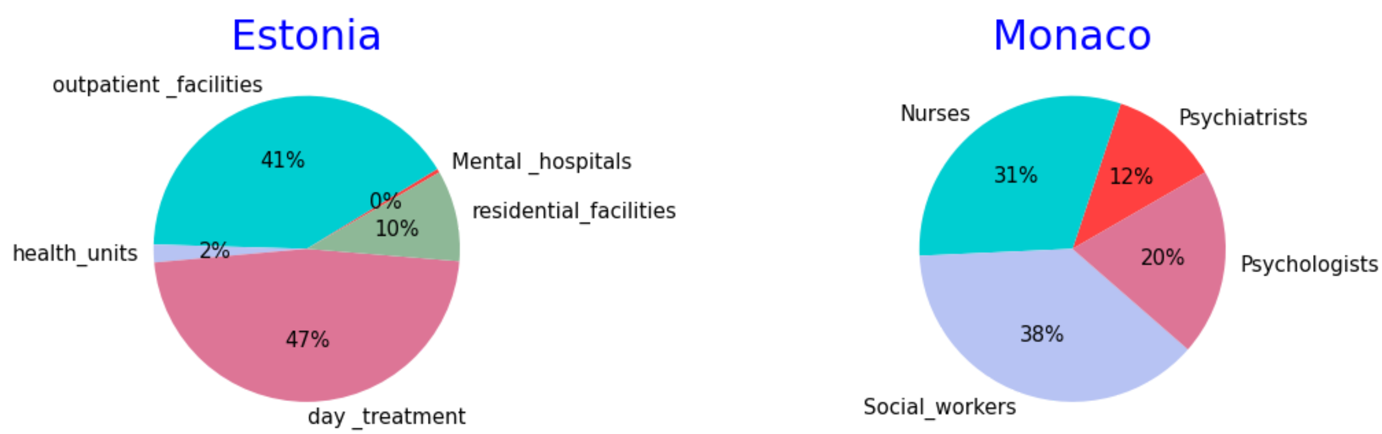


Figure 14: Percentage Human Resources and Facilities across Continents

In [81]:
```
lineGraphContinentalHRFac()
```

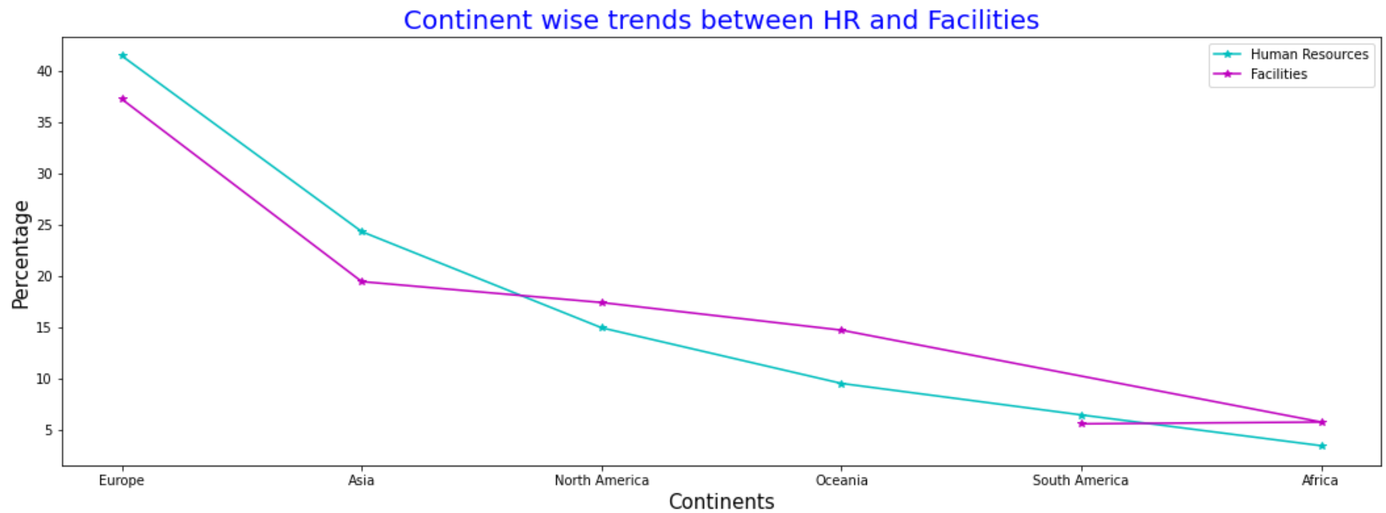
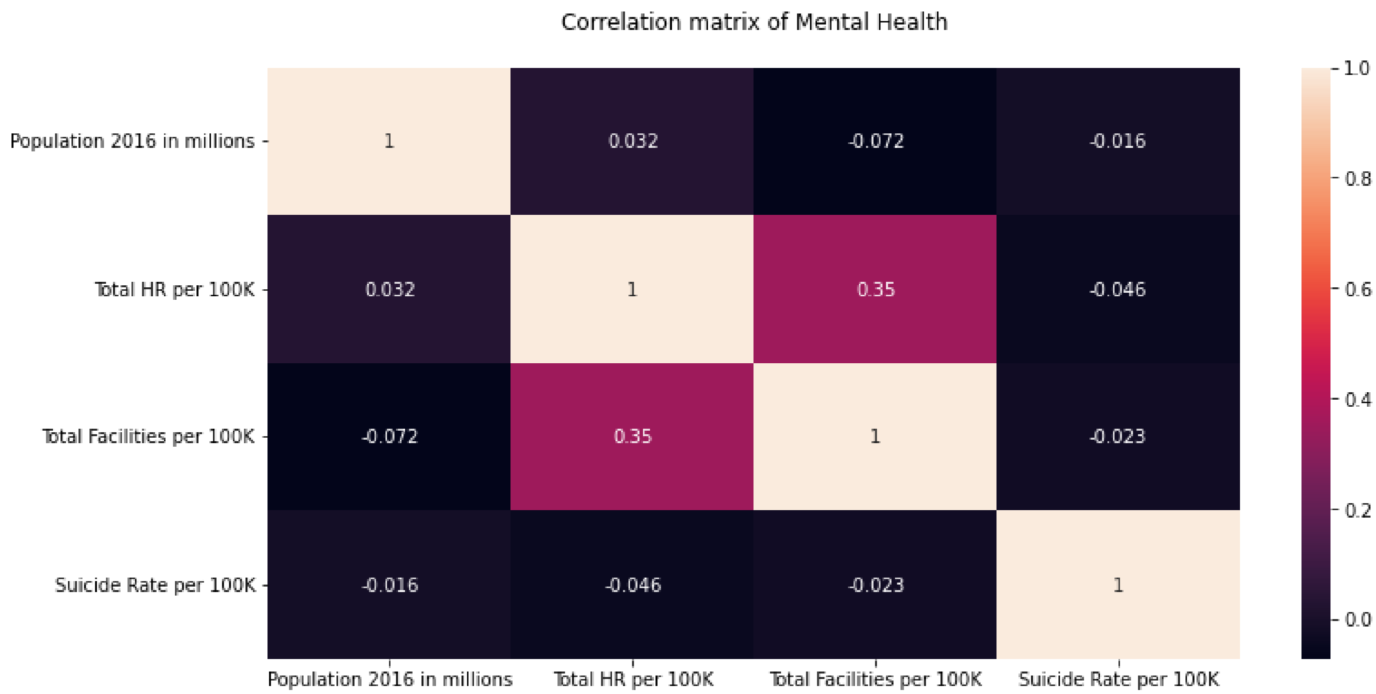


Figure 15: Correlation Matrix

In [82]:
```
corrMatrix()
```



Conclusion (5 marks)

Achievements

Our overall aim was to provide strategic insight into this extremely complex issue of global suicide rate and we can hypothesize that our detailed analysis certainly shed some light on worrying rates of global suicide. We have exemplified our analysis of interesting trends in various demographic information related to suicide rates worldwide and demonstrated through varied kinds of plots. This analysis highlights the importance of driving innovation in current mental health care and in bringing hope for the future.

Limitations

The project gives a snapshot of the overall suicide rate in the world and its associated factors, but only for the year 2016 instead of all 4 years. It does not provide the details of suicide methods or types of mental disorders leading an individual to take his/her own life. Even though suicide is claimed to be one of the leading causes of death globally, the current dataset lacks in providing the percentage of death by other causes for us to prove this claim analytically.

Future Work

In future work, we would like to obtain sundry other datasets where we can extend our current analysis to understand the type of mental disorders an individual experiences, if any, before taking the suicide decision. In addition to different age groups and genders, we would also like to include the occupation of each individual to comprehend which career place has the highest suicide rates. Based on this, we can predict the places where mental health care facilities or support groups can be enhanced to aid in suicide prevention and maintain a healthy environment.